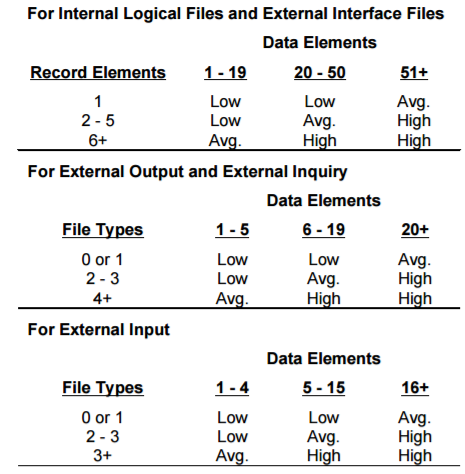
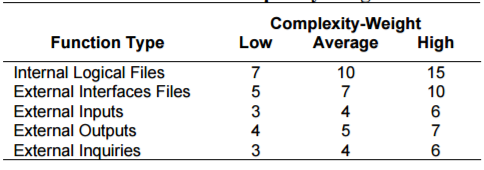
1. **Project size, cost and effort estimation**
   1. **FP estimation**
      1. **Complexity levels and function points**

The table below (from COCOMO II documentation) shows how to evaluate weights of Function Points, based on

the number of data elements:



According to these criteria, the next table (from COCOMO II documentation) shows the numeric weights for the functions:



* + 1. **Internal logic files**

An Internal Logic File (ILF) is a homogeneous set of data used and managed by the application.

In our system, we can consider the database tables where we store persistent information as ILFs; the tables are the following:

* User
* Reservation
* Car
* Supervisor
* Safe area
* Safe park slot

Except for the supervisor table, which is simple and will not contain many elements, the other tables will be a bit more complex and populated; so, we decide to assign the following complexities:

|  |  |  |
| --- | --- | --- |
| **ILF** | **Complexity** | **FP** |
| User | AVG | 10 |
| Reservation | AVG | 10 |
| Car | AVG | 10 |
| Supervisor | LOW | 7 |
| Safe Area | AVG | 10 |
| Safe park slot | AVG | 10 |

* + 1. **External interface files**

An External Interface File (EIF) is a homogeneous set of data used by the application but generated and maintained by other applications.

The only external application our system interact with is:

* Google Maps

The interface will manage a large amount of data regarding the position of cars and users and the requests to calculate paths and distances, so we choose a high level of complexity.

|  |  |  |
| --- | --- | --- |
| **ELF** | **Complexity** | **FP** |
| Maps | HIGH | 10 |

* + 1. **External input**

An External Input (EI) is an elementary operation to elaborate data coming from the external environment.

In our system, we can consider as EIs the following:

* User registration
* User login
* Request for car’s research
* Request for car’s reservation
* Request for car’s unlock
* Car’s data (from sensors of physical cars)

The information needed to perform a login, to request a reservation or to unlock a car, are few and trivial, so we decide to set the complexity for these functions at a low level.

For a new user registration and for the research of a car, the amount of data sent and the controls done on it are a bit more complex: we set the complexity at an average level.

Lastly, we need to manage data sent from the cars constantly, in order to maintain the internal status of our system consistent and coherent; considering the number of the cars and the amount of data coming from them, we set the complexity of the car’s data at a high level.

|  |  |  |
| --- | --- | --- |
| **EI** | **Complexity** | **FP** |
| User registration | AVG | 4 |
| User login | LOW | 3 |
| Request for car’s research | AVG | 4 |
| Request for car’s reservation | LOW | 3 |
| Request for car’s unlock | LOW | 3 |
| Car’s data | HIGH | 6 |

* + 1. **External output**

An External Output (EO) is an elementary operation that generates data for the external environment and it usually includes the elaboration of data from logic files.

In our system, the EOs are:

* Confirmation email for registration
* Results of car research
* Lock/unlock commands to physical car
* Payment requests
* Confirmation email for payment

Sending emails do not require extremely difficult computation, and the same is valid for sending lock/unlock commands: we set the complexity of these external output to a low level.

The function for searching car, on the other hand, could produce several results that have to be sent, and the computational effort is a bit higher. Also, the payment request function is not trivial, the system must monitor each active reservation and notice when one of them expires, then calculate the total amount and send the payment request to the bank. The complexity for these function is set to average.

|  |  |  |
| --- | --- | --- |
| **EO** | **Complexity** | **FP** |
| Confirmation email reg. | LOW | 4 |
| Results of car research | AVG | 5 |
| Lock/unlock commands | LOW | 4 |
| Payment requests | AVG | 5 |
| Confirmation email payment | LOW | 4 |

* + 1. **External Inquiry**

An External Inquiry (EQ) is an elementary operation that involves input and output, without significant elaboration of data from logic files.

In our system the EQs are:

* Notifications about reservation (to user and to physical car)
* Data requests from supervisors

Data requests from the supervisors are trivial queries, so we set the complexity level to low.

|  |  |  |
| --- | --- | --- |
| **EQ** | **Complexity** | **FP** |
| Notifications | AVG | 4 |
| Data request | LOW | 3 |

* + 1. **Final estimation**

We can now calculate the total amount of FP of the system:

|  |  |
| --- | --- |
| **Function type** | **FP** |
| Internal logic files | 57 |
| External interface files | 10 |
| External Input | 23 |
| External output | 22 |
| External inquiries | 7 |
| **TOTAL** | **119** |

According to the COCOMO II documentation, the multiplicator to convert the Function Points to Source Lines of Code for Java is 53, so we have:

**SLOC = 53 x 119 = 6307**

* 1. **COCOMO II Effort estimation**

In COCOMO II effort is expressed as Person-Months (PM). A person month is the amount of time one person spends working on the software development project for one month, considering 152 hours of work per Person-Month.

The COCOMO II effort estimation model is:

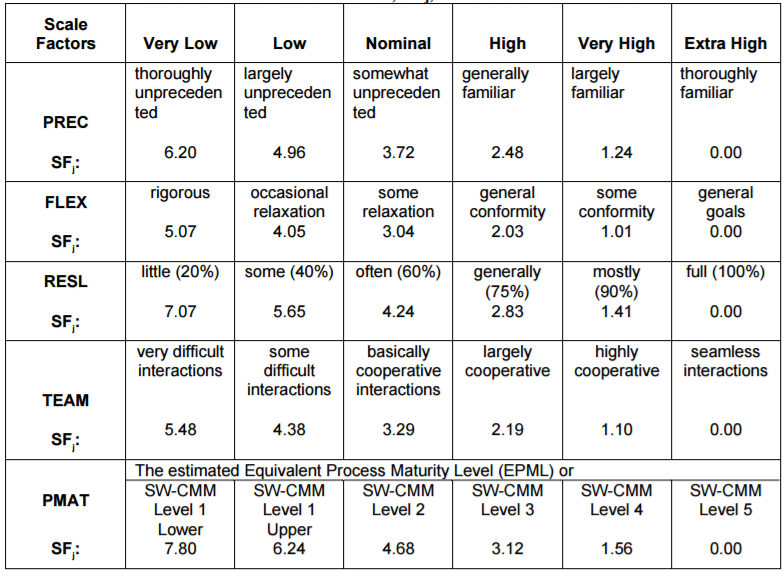


Where A = 2.94, and Size is the number of SLOC calculated before, given in thousands (SLOC/1000).

The parameters E and EM are calculated using respectively scale factors and cost drivers.

* + 1. **Scale factors**

The exponent E is an aggregation of five scale factors (SF) that account for the relative economies or diseconomies of scale:



**Precedentedness (PREC)**: reflects the previous experience of the organization with

this type of project.

The precedentedness is **Low** because some of us have little experience of software design but most of the notions are new to us.

**Flexibility (FLEX)**: reflects the degree of flexibility in the development process.

We set it to **Very High** because we have been given only the definition of the goals of the system, without non-functional or technological constraints.

**Risk resolution (RESL)**: evaluate the extent of risk analysis carried out.

We set it to **Nominal**, because a risk analysis was made and exposed in chapter 5.

**Team cohesion (TEAM)**: evaluate how well the development team know each other

and work together

We set it to **High**.

**Process maturity (PMAT)**: reflects the process maturity of the organization. We

set it at **High**, which corresponds to CMM Level 3.

The COCOMO II formula for E is:



Where B = 0.91

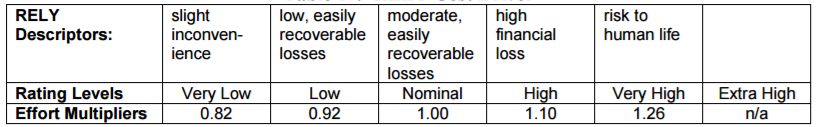
|  |  |  |
| --- | --- | --- |
| **SF** | **Factor** | **Value** |
| PREC | Low | 4.96 |
| FLEX | Very high | 1.01 |
| RESL | Nominal | 4.24 |
| TEAM | High | 1.10 |
| PMAT | High | 1.56 |
| **TOT** | | 12.87 |

**E = 0.91 + 0.01** X **12.87 = 1.0387**

* + 1. **Cost driver**

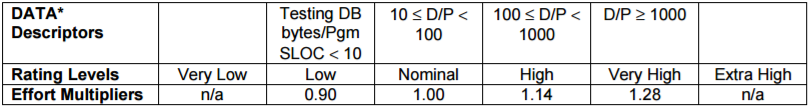
Cost drivers are used to capture characteristics of the software development that affect the effort to complete the project; Each rating level of every multiplicative cost driver has a value which is the effort multiplier (EM) used in the effort estimation formula.

We use the Post-Architecture model since we already have developed a detailed architecture for the system; the seventeen post-architecture effort multiplier are:

**Required Software Reliability (RELY):** This is the measure of the extent to which the software must perform its intended function over a period of time.

We set it to **low** because a failure won’t have critical consequences.

**Data base size (DATA):** This measure captures the effect that large test data requirements have on product development.



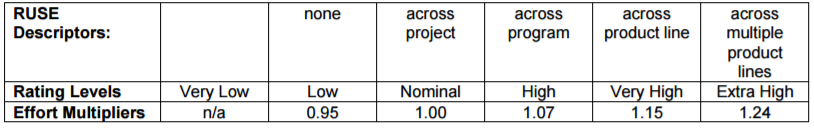
We set it to **Nominal** because we do not need extremely large amount of data to test the functionality of the system.

**Product Complexity (CPLX):** Complexity is divided into five areas: control perations, computational operations, device-dependent operations, data management operations, and user interface management operations.



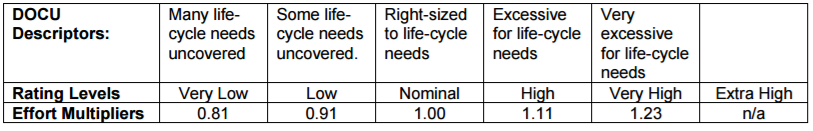
According with the COCOMO II CPLEX rating scale, we set this cost driver to **High.**

**Required Reusability (RUSE):** This cost driver accounts for the additional effort needed to construct components intended for reuse on current or future projects.



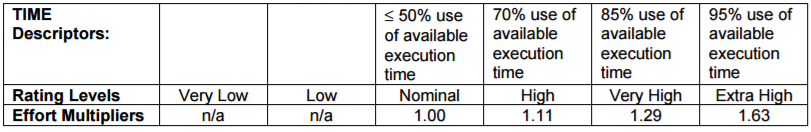
We choose a **Low** value because we assume that we do not need to reuse components from this project.

**Documentation match to life-cycle needs (DOCU):** thiscost driver is evaluated in terms of the suitability of the project’s documentation to its life-cycle needs.



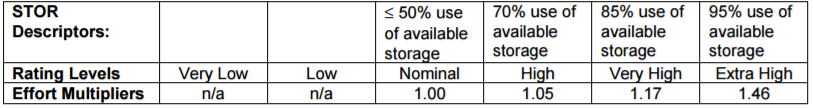
We set it to **Nominal** because the documentation developed for the project is sufficient to cover the its life cycle.

**Execution Time Constraint (TIME):** This rating is expressed in terms of the percentage of available execution time expected to be used by the system consuming the execution time resource.



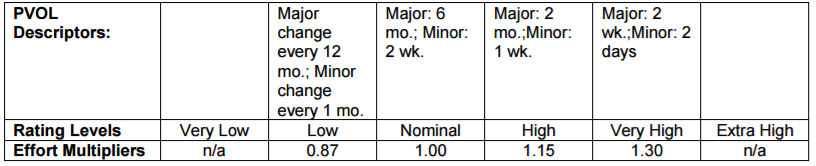
We set it to **Nominal.**

**Main Storage Constraint (STOR):** This rating represents the degree of main storage constraint imposed on a software system or subsystem.



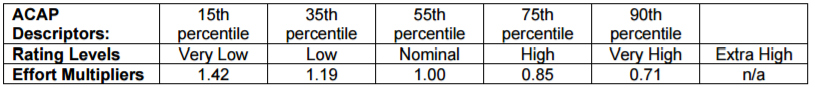
We choose a **Nominal** value.

**Platform Volatility (PVOL):** This rating ranges from low, where there is a major change every 12 months, to very high, where there is a major change every two weeks.



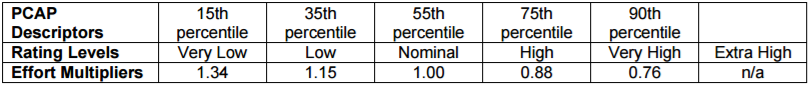
We set it to **Low.**

**Analyst Capability (ACAP):** 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.



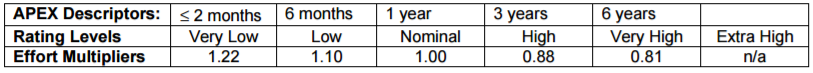
We choose a **Nominal** value because we are not considering our level of experience but only the quality of the design work.

**Programmer Capability (PCAP):** Major factors which should be considered in the rating are ability, efficiency and thoroughness, the ability to communicate and cooperate and the ability to deal with COTS packages.



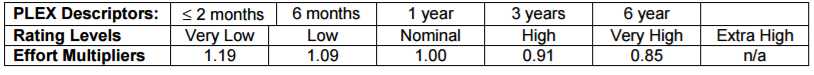
We set it to **Nominal.**

**Application Experience (APEX):** The rating for this cost driver is dependent on the level of applications experience of the project team developing the software system.



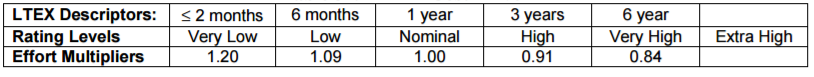
We choose a **Very Low** value for this cost driver because we are relatively new and unexperienced in this sector of applications.

**Platform Experience (PLEX):** this cost driver reflects the importance of understanding the use of more powerful platforms.



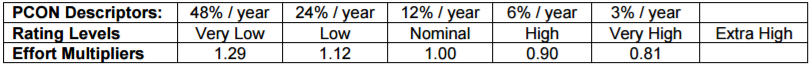
We set this driver to a **Low** value due to our limited knowledge of the enterprise platforms.

**Language and Tool Experience (LTEX):** This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem.



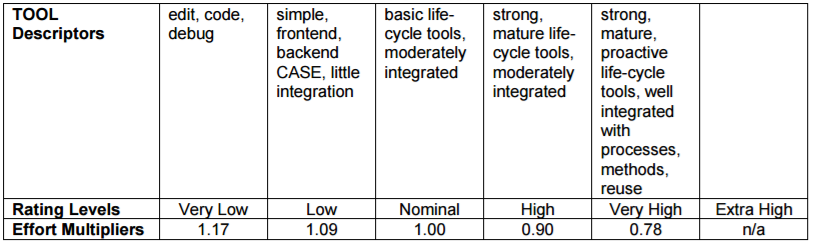
We set it to **Nominal.**

**Personnel Continuity (PCON):** This rating scale is in terms of the project’s annual personnel turnover.



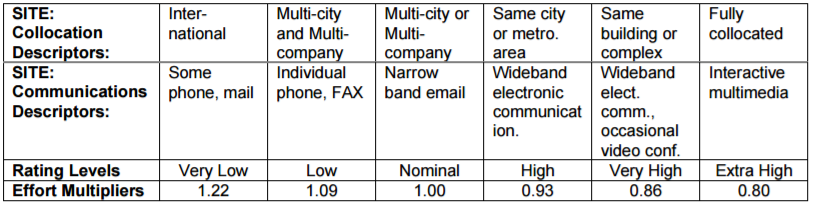
We obviously choose the **Very High** value because our team is fixed.

**Usage of Software Tools (TOOL):** The tool rating ranges from simple edit and code, very low, to integrated life-cycle management tools, very high.



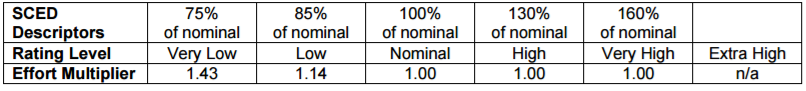
We set it to **Nominal.**

**Multisite Development (SITE):**

****

We set this cost driver to **Nominal** because the team members live in different cities and the work was often done from home.

**Required Development Schedule (SCED):** 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.



We set it to **Nominal** because we managed to follow the given deadline correctly without rushing or procrastinate.

|  |  |  |
| --- | --- | --- |
| **Cost driver** | **Factor** | **Value** |
| RELY | Low | 0.92 |
| DATA | Nominal | 1.00 |
| CPLX | High | 1.17 |
| RUSE | Low | 0.95 |
| DOCU | Nominal | 1.00 |
| TIME | Nominal | 1.00 |
| STOR | Nominal | 1.00 |
| PVOL | Low | 0.87 |
| ACAP | Nominal | 1.00 |
| PCAP | Nominal | 1.00 |
| APEX | Very low | 1.22 |
| PLEX | Low | 1.09 |
| LTEX | Nominal | 1.00 |
| PCON | Very High | 0.81 |
| TOOL | Nominal | 1.00 |
| SITE | Nominal | 1.00 |
| SCED | Nominal | 1.00 |

**∏**EM = 0.9583

* + 1. **Effort estimation**

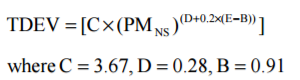
We can now calculate the Person-Month estimation:



PM = 2.94 X 6.3071.0387 X 0.9583 = **19.08 person-month**

* + 1. **Schedule estimation**

The COCOMO II formula to estimate the project duration is:



From the formula above, we obtain an estimated duration of **9.04** months with an optimal number of team member (2.11); since our team is composed by 3 person, a duration estimation could be: 19.08/3 = 6.36 months, which we can approximate to **7 months**

Bibliography

http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII\_modelman2000.0.pdf