POLITECNICO
MILANO 1863

POWER ENJOY

Project Plan Document

Lorenzo Casalino - 877421
Tommaso Castagna

Document version 1.0

Contents

1 Introduction **1**
1.1 Revision History . . . 1
1.2 Purpose . . . 1
1.3 Scope . . . 1
1.4 Definitions, Acronyms and Abbreviations . . . 1
1.5 Reference Documents . . . 1

2 Project Size, Effort and Cost Estimation **2**
2.1 Size Estimation . . . 2
2.1.1 Internal Logic File . . . 3
2.1.2 External Interface File . . . 4
2.1.3 External Input . . . 5
2.1.4 External Inquiries . . . 7
2.1.5 External Output . . . 8
2.2 Effort and Cost Estimation . . . 9
2.2.1 Scale Drivers . . . 9
2.3 Cost Drivers . . . 11

1 Introduction

1.1 Revision History

The history of document revisions is here recorded in tabular format, mapping the document version with the changes brought to document itself.

The current version of the document is highlighted by the version number in bold format.

Version	**Revision**
1.0	First released version.

1.2 Purpose

1.3 Scope

1.4 Definitions, Acronyms and Abbreviations

1.5 Reference Documents

2 Project Size, Effort and Cost Estimation

In this section the estimation process of the three key aspects for an effective project planning, namely size, effort and costs expected, is described in details, pointing out the rationale of each single step of the process itself.

The size estimation process is led by a **functionality-provided** based approach, whereby the estimation is made according to the functionalities that the software product is planned to provide. To support this strategy, *Function Points* technique is used.

Regarding the effort and cost estimation, the process is based on **algorithmic approach**, that is the use of an algorithmic model based on a simple equation which output depends on several factors regarding the project. The algorithmic model here used is based on *COCOMO II*.

2.1 Size Estimation

As explained in the introductory paragraph to this section, the size estimation effort is based on the estimation of the so-called *Function Points*. Function points are a statistical method of estimate the size of a software project evaluating the different functionalities provided by the software product in exam.

According to this approach, functionalities are divided into 5 **function types**, or category:

- **Internal Logical File.**
- **External Logical File.**
- **External Input.**
- **External Output.**
- **External Inquiry.**

For each of these type, a **weight** is associated. These weights are statistically determinated and vary according to the **complexity** of the function type. The complexity of a function type can be derivided consulting the related *rating tables*.

	Data elements		
RET	$1 \rightarrow 19$	$20 \rightarrow 50$	51+
1	Low	Low	Average
$2 \rightarrow 5$	Low	Average	High
6+	Average	High	High

In order to retrieve the number of function points assigned to the software product given a specific function type, a simple equation is applied:

$$PFP_t = N_t * FP_t, t \in T = [ILF, ELG, EI, EO, EINQ]$$

This equation returns the *partial function points* obtained by multiplying the number of functionalities of a certain catergory for the weight associated to that category.

The total number of function points assigned to the whole project is computed by the *Unadjusted Function Points* equation, which simply compute the sum of each PFP previously calculated.

$$UFP = \sum_t FPF_t, t \in T$$

2.1.1 Internal Logic File

	Data elements		
RET	$1 \rightarrow 19$	$20 \rightarrow 50$	51+
1	Low	Low	Average
$2 \rightarrow 5$	Low	Average	High
6+	Average	High	High

- **User account informations**: this file contains the informations that any user is asked to type at registration time to the service. The informations (Name, surname, password, email, telephone number, postal code, city, birthday, driving license, payment informations) are recorded inside one type of record.
- **Operator account informations**: the purpose is similar to the logic file storing the user account informations, but with the difference of not containing the driving license and the payment informations. Only one type of record is exploited.
- **Priviledges**: to avoid the execution of actions not allowed to some category of users, this file is used to map the priviledge level associated to each account category. Only one record type is used.
- **Parking area informations**: area informations, such as area identifier and coordinates describing the geographical boundaries of the parking area, are stored in this internal file, by means of one record type.
- **Safe parking area informations**: extension of the previous internal logic file, which adds the presence of the total number of parking spots inside the area.

- **Vehicle informations**: this file contains both the usual informations characterizing a vehicle (Plate number, frame number, model, matriculation date, fuel type, shift type) and the informations regarding the actual status of the vehicle (Fuel percentage, availability, position). Two records are used to store, respectively, the type of informations described above.
- **Reservation informations**: the informations regarding each reservation is stored by the support of three records. The first record stores an identificator to the reservation, the begin and end date of the reservation, the identificator of the reserving user, the plate number of the reserved vehicle and the total charge. The second structure is used to keep track of events that trigger a policy rule. An identificator to the event, the identificator of the policy rule, event date, event condition and effect are stored. The third record is used to map a reservation to one or more event.
- **Policy rules**: to implement the detection of good or bad behaviours, according to the policy adopted by the car sharing society, the rules are encoded and stored in an appropriated format. This logic file serves this purpose, using one record structure to store the rule identificator, the encoded conditions and encoded effects.
- **Mantainance tasks**: mantainance tasks assigned to a certain operator are stored by this logic file, using two record type. The first used to store the identificator task and its description, while the second is used as a support, mapping tasks to a specific operator using his/her identificator.

Making reference to the ILF rating table, we can evaluate the partial function point value:

ILF	**Complexity**	**Weigth**
User account informations	Low	7
Operator account informations	Low	7
Priviledges	Low	7
Parking area informations	Low	7
Safe parking area informations	Low	7
Vehicle informations	Low	7
Reservation informations	Low	7
Policy rules	Low	7
Mantainance tasks	Low	7
Partial Function Point	63	

2.1.2 External Interface File

The external services used by our system are developed by third parts. Consequently, it's quite hard to understand how many record types and data elements of each external file are used. In order to perform a meaningful estimations, the following assumptions about the external services are taken in consideration.

	Data elements		
RET	1 → 4	5 → 15	15+
¡ 2	Low	Low	Average
2	Low	Average	High
¿ 2	Average	High	High

- **Driver licenses validator web service files**: we assume that the web service needs only to use informations regarding stored driver licenses and drivers informations, hence use of two record structure types.
- **Payment system web service files**: as for the previous web service, it's reasonable think that the payment web service relies only the payment informations and payment service's customers informations.
- **Google maps web service files**: the kind of informations needed by this service are unknown to us, but we assume that the operations performed for map plotting and so on involves a complex external file.

EIF	**Complexity**	**Weight**
Driver licenses validator web service	Low	5
Payment system web service	Low	5
Google maps web service	High	10
Partial Function Point	20	

2.1.3 External Input

- **Login**: elementary operation that involves the user account informations (or the operator account informations).
- **Logout**: elementary operation that does not require the involvement of any internal or external file to achieve the result.
- **Register account**: functionality that relies on the user account or the operator internal logic file.
- **Delete account**: operation relying only on user account or operator account internal logic file.
- **Update user account informations**: this operation is more complex with respect to the last two since modification of a user information can involves other internal or external files. Indeed, for instance, the payment informations editing needs the intervention of the payment web service in order to verify the validity of the updated informations, hence the involvement of the payment system external file. This external input relies on the user account internal file, the payment system web service and driver license web service external files.

- **Update operator account informations**: with respect to the update operation regarding the user account informations, this process is simpler since only the operator informations ILF is involved.
- **Create reservation**: the creation of a reservation from a user is a quite simple operation, involving only the reservation and the vehicle informations internal logic files.
- **Terminate reservation**: complex operation that need the support of the reservation informations, vehicle informations internal logic file and the payment web service external file.
- **Update reservation**: updating the information regarding an already terminated reservation is a quite simple operation that involves the reservation informations.
- **New vehicle insertion**: simple operation relying only on the vehicle informations internal logic file.
- **Vehicle deletion**: simple operation relying only on the vehicle informations internal logic file.
- **Vehicle informations update**: simple operation where vehicle informations internal file is used.
- **Vehicle unlock**: operation not involving any internal or external file.
- **Rule policy activation**: whenever the driver performs a particular action described into the policy rules, the software system reacts applying the related effects to the reservation. Reservation informations and policy internal logic files are involved.
- **Insert new policy rule**: operation through which the policy rules information file is modified.
- **Remove policy rule**: elementary operation involving only the policy rules information file.
- **Update policy rule**: simple operation involving the policy internal file.
- **Insert parking area**: the insertion of a new parking area, either a normal area or a safe area, is an operation which involves the parking area and the safe parking area internal files.
- **Remove parking area**: basic operation involving the parking area and the safe parking area internal files.
- **Update parking area informations**: same complexity and same internal files involved as for the previous two parking area-related functionalities.
- **Insert new task**: new tasks to be assigned to an operator is a simple operation where only the mantainance task informations file is involved.

- **Remove task**: simple functionality modifying the mantainance task information file.
- **Update task**: simple operation involving solely the mantainance task information file.

EI	**Complexity**	**Weigth**
Login	Low	3
Logout	Low	3
Register account	Low	3
Delete account	Low	3
Update user account	Average	4
Update operator account	Low	3
Create reservation	Low	3
Terminate reservation	Average	4
Update reservation	Low	3
Vehicle insertion	Low	3
Vehicle deletion	Low	3
Vehicle update	Low	3
Vehicle unlock	Low	3
Rule policy activation	Average	4
Insert policy rule	Low	3
Remove policy rule	Low	3
Update policy rule	Low	3
Insert parking area	Low	3
Remove parking area	Low	3
Update parking area	Low	3
Insert task	Low	3
Remove task	Low	3
Update task	Low	3
Partial Function Point	72	

2.1.4 External Inquiries

- **Retrieve reservation history**: the retrieve of the full reservation history (or part of it) is an operation that involves only two internal logic files, namely the reservation informations and the vehicle informations ILF.
- **Retrieve account informations**: operation involving either the user account informations ILF or the operator ILF.
- **Retrieve vehicle list**: simple operation involving the vehicle informations ILF.
- **Retrieve vehicle informations**: the characteristic informations regarding a vehicle are retrieved inquiring solely the vehicle informations ILF.

- **Retrieve vehicle status**: the informations regarding the current status of a specific vehicle are stored into the vehicle informations ILF.
- **Retrieve current reservation informations**: operation similar to the reservation history retrive operation. The same ILF are involvedo.
- **Retrieve area list**: simple inquire involving the area informations ILF.
- **Retrieve specific area informations**: as the previous operation, this one is simple as well and involves the same ILF.
- **Retrieve mantainance operator list**: only the operator ILF is involved in this operation.
- **Retrieve mantainance task list**: simple inquire operations regarding the mantainance task informations ILF.
- **Retrieve policy list**: the retrieval of the total list of policy rules present into the system or part of them is a simple operation involving the policy ILF.
- **Retrieve policy rule informations**: the informations regarding a specific policy rule involves the querying of the policy ILF.
- **Retrieve vehicles in specific area**: the achievement of this operation is reached through the querying of the vehicle informations ILF.

IQ	**Complexity**	**Weigth**
Retrieve reservation history	Low	3
Retrieve account informations	Low	3
Retrieve vehicle list	Low	3
Retrieve vehicle status	Low	3
Retrieve current reservation info	Low	3
Retrieve area list	Low	3
Retrieve specific area info	Low	3
Retrieve mantainance operator list	Low	3
Retrieve mantainance task list	Low	3
Retrieve policy rule list	Low	3
Retrieve policy rule info	Low	3
Retrieve vehicle in specific area	Low	3
Partial Function Point	36	

2.1.5 External Output

- **Notify reservation expired**.
- **Notify policy rule(s) application**.
- **Notify registration result**.

- **Notify login result**.
- **Notify logout result**.
- **Notify insertion result**.
- **Notify deletion result**.
- **Notify update result**.
- **Notify reservation request result**.
- **Notify unlock request result**.

EO	**Complexity**	**Weigth**
Notify reservation expired	Low	3
Notify policy rule(s) application	Low	3
Notify registration result	Low	3
Notify login result	Low	3
Notify logout result	Low	3
Notify insertion result	Low	3
Notify deletion result	Low	3
Notify update result	Low	3
Notify reservation request result	Low	3
Notify unlock request result	Low	3
Partial Function Point	30	

Applying the equation above shown, the unadjasted function point value is equal to 221 points. To give much flexibility as possible to the project, the gearing factors used to estimate the lower bound and the upper bound are calculated as the mathematical average between the gearing factors of J2EE and .NET, the two most widespread technologies.

Lower bound : 52 * 221 = 11.492 kilo-lines of code Upper bound : 64 * 221 14.144 kilo-lines of code

2.2 Effort and Cost Estimation

2.2.1 Scale Drivers

The scale drivers are five distinct factors used to translate directly in the effort estimation process specific characteristics that have a huge relevance and impact on the project development.

To each scale driver can be assigned a value raging from *very low* to *very high*. The traslation in the effort estimation computation is performed associating to each scale driver's value a decimal value obtained through statistical analysis.

These five scale drivers are:

- **Precedentedness**: this factor measures the degree of familiarity that the development team has regarding the project under analysis.

  Our experience with this kind of projects is really limited, therefore a *Low* value is assigned.

- **Development flexibility**: this factor reflects the degree of flexibility allowed for the development of the project. This factor is computed taking in consideration the presence of pre-enstablished requirements and software conformance with external interface specification.

  Our projects involves some pre-defined requirements but no external interface specifications are required, hence the value of this scale driver is set to *nominal*.

- **Architecture/Risk resolution**: scale factor measuring the quality of the risk management plan and scheduling/budget compatibility with the latter.

  The risk management plan defined for this project embrace and define a recovery plan for almost every predictable risks, respecting the scheduling and the budget defined. *High* value is assigned.

- **Team cohesion**: scale factor measuring reflecting the degree of cooperation among the team members. Our team had divergence on project choices several time that brought a slow down in the project development. A *Nominal* value is assigned.

- **Process maturity**: factor measuring the degree of maturity reached by the organization's development processes. Since this is our first project, is quite hard to assess the maturity of our approach. Hence, even if the applied processes to the development could be typical of a level-3 organization, we set the factor's value to *level 2*, typical of processes designed for specific projects.

Refering to the scale drivers table, the following table is produced

EO	**Complexity**	**Weigth**
Precedentedness	Low	4.96
Development flexibility	Nominal	3.04
Architecture/Risk resolution	High	1.41
Team cohesion	Nominal	3.29
Process maturity	Level 2	4.68
Total value	17.38	

The *total scale factor E* has a value computable by mean of the following equation

$$E = B + 0.01 * \sum_{f} (SF_f) = 1,0838$$

with $B = 0.91$ and f representing one of the scale drivers.

2.3 Cost Drivers

Similar to the scale factors, the cost drivers are a set of factors describing in terms of cost the impact that they have on the whole project. Each cost driver has a impact level, raging from *very low* to *very high*, and each impact level has a statistical decimal value associated.

The set of cost drivers vary according to the architectural state of the project, which may be in a*Post-architectural* state or *Early design* state. Post-architectural's cost drivers may rely on more precise and detailed informations about the project, producing more precise estimates. The early design model, instead, due to the lack of informations typical of early project provides less accurate figures regarding the effort estimation.

As specified in the *introductive paragraph* to this section, our project follow a early design model. The set of cost drivers used for the early design model is derived by the combinations of different cost drivers defined for the post-architectural model. In the following, the cost drivers for the post-architectural model are described along with the rationale for the assigned value. At the end of this analysis, the early design cost drivers values are computed and shown.

Early design cost drivers	**Post-architecture cost drivers**
Personell Capability [PERS]	ACAP PCAP PCOPN
Product Reliability and Complexity [RCPX]	RELY DATA CPLX DOCU
Developed for Reusability [RUSE]	RUSE
Platform Difficulty [PDIF]	TIME STOR PVOL
Personnel Experience [PREX]	APEX PLEX LTEX
Facilities [FCIL]	TOOL SITE
Required Development Schedule	SCED

- **Required Software Reliability [RELY]**: this cost driver is a measure of the reliability required for the software to be developed. The value is assigned according to the level of danger and/or damage the interruption of the service may cause. Since the software system is designed to support a car-sharing service, a malfunction can lead to important finantial losses for the society. Hence the assigned value is set to *High*.
- **Database Size [DATA]**: the dimension of test data required for the software testing has an impact on the cost of development for the test. In particular, the database size required to store these test data is taken in consideration, evaluating the derived impact on the costs as the ration between the database size D and the LOSC P.

  Being in an early stage phase of the development, it's not easy to estimate the dimension of the database. We can assume a minimal dimension of 1 gigabytes. Therefore, the value of this cost driver is set to *Very High.*

- **Product Complexity [CPLX]**: this cost driver is divided into five area, shown in the table below, and its value is the average of the five area's values. The CPLX's value is set to *High.*

- **Developed for reusability [RUSE]**: this driver reflect the degree of reusability adopted for the implementation of system's components. No reusability feature with other projects is expected to be adopted in the design and implementation of the components, hence the cost driver's values is equal to *Nominal.*

- **Documentation Match to Life-Cycle Needs [DOCU]**: this cost driver measure the degree of suitability of the required documentation to the project's life-cycle. Since we're following a precise model of development which requires the appropriate documentation for each step, the documentation is right-sized to the project's life-cycle. The value is set to *Nominal.*

- **Execution Time Constraints [TIME]**: this factor estimates the expected amount of execution time used by the software system. The software system is expected to be deployed in a highly concurrent environment, with several requests. Therefore, the value of this driver is set to *Very High.*

- **Main Storage Constraints [STOR]**: this rating represent the percentual amount of main storage expected to be used by the system. We expect that the high number of generated and stored data won't take less than the 70% of the total storage available in the system. The cost driver's value is set to *High.*

- **Platform Volatility [PVOL]**: this driver measures the average time of hardware/software replacement and update expected. The software system is supposed to be deployed on an external cloud computing service. Major changes are expected to be applied to the software part or to the hardware living in the vehicles and performed quite rarely.

- **Analyst Capability [ACAP]**: cost driver that measure the analysis and design ability of the team regarding the project in analysis. It's the first time we're faced with a project of this size and complexity. Therefore, the assigned value to this factor is *Nominal.*

- **Programmer Cability [PCAP]**: driver analyzing the efficiency and ability of the programmers involved in the project development.Since we are in a early stage phase, the project is not developed yet. Hence, we can only estimate this parameter with respect to our experience, which is quite limited with this kind of projects. The value of the driver is set to *Nominal.*

- **Personnel Continuity [PCON]**: this driver measure the percentage of time that team members are unavailable for the project development. In our case, the driver is set to a *Very Low* value due to limited time we can spent for the project.

- **Application Experience [APEX]**: rating describing the experience of whole team with this kind of projects. As stated in the *PCAP driver* description, our experience is quite limited, hence the value for this driver is *Very Low.*
- **Platform Experience [PLEX]**: cost driver assessing the experience of the team with the platform used for the development. The project is in a early stage development, therefore the development platform is not decided yet. However, due to our limited experience with any development platform, we can assign a value of *Low* to this driver.
- **Language and Tool Experience[LTEX]**: experience and familiarity of the development team with he technology used, both languages and software tools, are describe by this driver. Most of the technologies and frameworks needed to achieve the goals of each step of the project's life cycle are quite unknown. Therefore, the set value for this driver is *Low.*
- **Use of Software Tools [TOOL]**: this driver attempts to assess the kind of tools used and their integration. The tools we expect to use are quite integrated and support quite well the project's life-cycle. The driver's value is set to *High.*
- **Multisite Development [SITE]**: the characteristics evaluated by this driver are the collocation of the site development and the communication channels used to manage the development activity. Our team is collocated in two different cities (Multi-city collocation) and uses several communication channels (VoIP technologies, text messages, emails etc...). Hence, the rating of this driver is set to *High.*
- **Required Development Schedule [SCED]**: this cost driver aims to rate the impact that the time constraints imposed to the project's team has on the whole project. this cost driver aims to rate the impact that the time constraints imposed to the project's team has on the whole project. The schedule assigned for this project is felt slightly compressed by the project's team, therefore the rate of this driver is set to *Low.*

  The PM's lower bound is 108.559   109. The PM's upper bound is 135.957 136.

  The scheduling activity has a lower bound $= 3.67 * 108^{0}.31476 = 16.02$ months and an upper bound $= 3.67 * 136^{0}.31476 = 17.23$ months.

Early design cost drivers	**Rating level**	**Effort multiplier**
PERS	Low	1.26
RCPX	Very High	1.91
RUSE	Nominal	1.00
PDIF	High	1.29
PREX	Very Low	1.33
FCIL	High	0.87
SCED	Low	1.14
Total Effort Multiplier		4.095