



POLITECNICO DI MILANO

A.A. 2015-2016

SOFTWARE ENGINEERING 2: “myTaxiService”

Project Plan Document

Andrijana Mirchevska (838622)

Marija Mavcheva(838647)

24 January 2016

Table of Contents

1. Introduction	3
2. Function Points Approach.....	4
3. COCOMO Approach	8
3.1. Estimated Calculations using COCOMO II	8
3.2. Precise Calculations using COCOMO II.....	9
3.3. Cost Drivers and Scale Drivers	11
3.3.1. Software Cost Drivers	11
3.3.2. Software Scale Drivers	13
4. Identifying the tasks and their schedule.....	14
5. Allocation of members to tasks	15
6. Risks and recovery actions.....	16
7. References	18

1. Introduction

In this document we will evaluate the time and the resources that are necessary for the development of the MyTaxiService application.

In order to do the analyses for the project, the Function Points technique is used and the results have been compared with the dimension of the project.

Also COCOMO approach is used to evaluate the effort for the implementation of MyTaxiService application, and the results have been compared with the time actually spent.

2. Function Points Approach

The Function Points is a technique that allows to evaluate the effort needed for the design and implementation of a project. We decided to use this approach also for our project, in order to evaluate the application dimension that is based on the functionalities of the application.

The list of functionalities is obtained from the RASD, and they're grouped into:

Internal Logic File (ILF) represents a homogeneous set of data used and managed by the application. In MyTaxiService application this corresponds to the Database table.

External Interface File (EIF) represents a homogeneous set of data used by the application but generated and maintained by other applications. In our case this corresponds to the two external platforms: Google Maps API and Mailing Services API.

External Input is an elementary operation to elaborate data coming from the external environment.

External Output is an elementary operation that generates data for the external environment.

External Inquiry is an elementary operation that involves input and output operations.

Every FPA (Function Point Analysis) must begin with grouping the components of the system we are analyzing. That's why the five groups (listed above) are distinguished. Once the components are selected and grouped, we can turn to analyzing itself.

Basically people solve problems by dividing them into smaller parts. And so is with measuring the application complexity with Function Points Method. So, instead of trying to evaluate the application as a whole, we need to rate each of the selected groups.

We need to classify the complexity of each category. We therefore have three possibilities - the complexity could be simple, medium, or complex. Then, the thing is to count the scores following the rules tabled below:

Function Type	Complexity		
	Simple	Medium	Complex
Internal Logic File (ILF)	7	10	15
External Interface File (EIF)	5	7	10
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6

So the number of Function Points is computed as the weighted sum of function types using the coefficients of the table above. We perform the calculations step by step:

- **Internal Logic Files (ILFs):** MyTaxiService application includes a number of ILFs that will be used to store the information about users, drivers, requests, reservations, reports, and notifications.

So the internal logic files users, drivers, reports and notifications have simple structure as it is composed of a small number of files, thus they adopt simple weight. And the number of function points is: $4 \times 7 = 28$ FPs.

The requests and reservations internal logic files have a more complex structure, which means they adopt medium weight. The number of function points is: $2 \times 10 = 20$ FPs.

The total FPs concerning ILFs is $28 + 20 = 48$ FPs.

- **External Interface Files (EIFs):** Our application has two EIFs, and those are the APIs for Google Maps and Mailing Services.

Google Maps external interface file has a complex structure and it adopts a complex weight. Its number of function points is: $1 \times 10 = 10$ FPs.

The Mailing Service external interface file adopts a simple weight. Thus its number of function points is: $1 \times 5 = 5$ FPs.

The total number of FPs concerning EIFs is $10 + 5 = 15$ FPs.

- **External Inputs:** The application interacts with the user to allow him/her to:
 - *Register*: this is a simple operation and it adopts the simple weight: $1 \times 3 = 3$ FPs.
 - *Login/Log out*: these are simple operations and they adopt the simple weight: $2 \times 3 = 6$ FPs.

- *Change email, change password, change avatar*: these are simple operations and they adopt the simple weight: $3 \times 3 = 9$ FPs.
- *Request a taxi/make reservation*: these are complex operations since they include interaction among the following components: user, request/reservation, taxi driver, and google maps API. And they adopt the complex weight: $2 \times 6 = 12$ FPs.
- *Cancel reservation*: this operation involves interaction among the following components: user, reservation, taxi driver, and google maps API. But still is of medium structure so it adopts the medium weight: $1 \times 4 = 4$ FPs.
- *Report user*: this operation involves interaction among the following components: user, administrator, and taxi driver. It is of medium structure so it adopts the medium weight: $1 \times 4 = 4$ FPs.
- *Confirm/decline a taxi call*: these operations involve interaction among the following components: taxi driver, request/reservation, and user. It is of medium structure so it adopts the medium weight: $2 \times 4 = 8$ FPs.
- *Managing requests/reservations*: this is a simple operation and it adopts the simple weight: $2 \times 3 = 6$ FPs.
- *Ban user*: this operation involves interaction among the following components: user, administrator, and taxi driver. It is of medium structure so it adopts the medium weight: $1 \times 4 = 4$ FPs.

The total number of FPs concerning EI is $3 + 6 + 9 + 12 + 4 + 4 + 8 + 6 + 4 = 56$.

○ **External Outputs:**

- After making a request/reservation the application will send to the user a notification in form of confirmation for the ride. It includes interaction among user, request/reservation, and taxi driver. So it is of complex structure and it adopts complex cost: $1 \times 7 = 7$ FPs.
- After making a reservation, the application will notify the user at least 2 hours before the ride takes place (since the user must make the reservation at least 2 hours before the ride). It includes interaction among user, request/reservation, and taxi driver. So it is of complex structure and it adopts complex cost: $1 \times 7 = 7$ FPs.

The total number of FPs concerning EO is $7 + 7 = 14$.

○ **External Inquiries: None**

- **Total FP number and summary:** We have computed the following value for the unadjusted FPs: $48 + 15 + 56 + 14 + 0 = 133$. This value can be used to estimate the effort in case we have some historical data that tell us how much time we usually take for developing a FP. Otherwise, it can be used as a basis to estimate the size of the project in KLOC and then use another approach such as COCOMO to estimate the effort.

3. COCOMO Approach

3.1. Estimated Calculations using COCOMO II

To calculate the average SLOC by using the FPs that are calculated above, we've taken the average conversion factor given at <http://www.qsm.com/resources/function-point-languages-table>, which is 46. This is an updated version that adds J2EE of the table included in official manual for COCOMOII.

$$133 \text{ FPs} * 46 = 6118 \text{ SLOC}$$

We consider the project with all "Nominal" (i.e., normal) Cost Drivers and Scale Drivers would have an EAF of 1.00 and exponent, E, of 1.0997. So we calculate the effort with the following formula:

$$\text{effort} = 2.94 * \text{EAF} * (\text{KSLOC})^E$$

and we obtain:

$$\text{effort} = 2.94 * (1.0) * (6.118)^{1.0997} = 21.5 \text{ Person/Months}$$

Assuming that the project is expected to consist of 6118 source lines of code, COCOMO II estimates that 21.5 Person/Months of effort is required to complete it.

EAF - Effort Adjustment Factor derived from Cost Drivers

E - Exponent derived from Scale Drivers

Now we calculate the duration (schedule) of the project in month, which is based on the effort predicted by the effort equation:

$$\text{Duration} = 3.67 * (\text{effort})^{SE}$$

We consider the exponent SE of 0.3179 (calculated from new scale drivers for schedule) and we obtain:

$$\text{Duration} = 3.67 * (21.5)^{0.3179} = 9.73 \text{ Months}$$

Also the number of people needed to complete the project is estimated. The following formula is used:


$$N_{\text{people}} = \text{Effort} / \text{Duration}$$

So we obtain:

$$N_{\text{people}} = 21.5 / 9.73 = 2.21 \sim 2 \text{ people}$$

3.2. Precise Calculations using COCOMO II

So in order to obtain more precise results for the COCOMO II model we used the online tool (<http://csse.usc.edu/tools/COCOMOII.php>). Below is the report of the site and the choice made about the Scale Drivers and Cost Drivers to obtain the results.



COCOMO II - Constructive Cost Model

Software Size Sizing Method | Source Lines of Code ▼

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	6118						
Reused	0	0	0				
Modified	0						

Software Scale Drivers

Precedentedness	High ▼	Architecture / Risk Resolution	High ▼	Process Maturity	Nominal ▼
Development Flexibility	Low ▼	Team Cohesion	Very High ▼		

Software Cost Drivers

Product		Personnel		Platform	
Required Software Reliability	Low ▼	Analyst Capability	Low ▼	Time Constraint	Nominal ▼
Data Base Size	Nominal ▼	Programmer Capability	High ▼	Storage Constraint	High ▼
Product Complexity	Low ▼	Personnel Continuity	Nominal ▼	Platform Volatility	Low ▼
Developed for Reusability	High ▼	Application Experience	Low ▼	Project	
Documentation Match to Lifecycle Needs	High ▼	Platform Experience	Low ▼	Use of Software Tools	High ▼
		Language and Toolset Experience	Low ▼	Multisite Development	Nominal ▼
				Required Development Schedule	Nominal ▼

Maintenance | Off ▼

Software Labor Rates
Cost per Person-Month (Dollars) 2000

Figure 1. Input data

Results

Software Development (Elaboration and Construction)

Effort = 21.5 Person-months

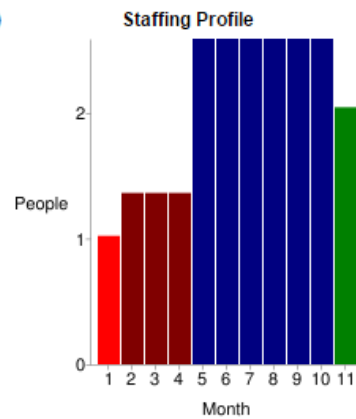
Schedule = 10.1 Months

Cost = \$43005

Total Equivalent Size = 6118 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.3	1.3	1.0	\$2580
Elaboration	5.2	3.8	1.4	\$10321
Construction	16.3	6.3	2.6	\$32684
Transition	2.6	1.3	2.0	\$5161



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

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.2	0.6	1.6	0.4
Environment/CM	0.1	0.4	0.8	0.1
Requirements	0.5	0.9	1.3	0.1
Design	0.2	1.9	2.6	0.1
Implementation	0.1	0.7	5.6	0.5
Assessment	0.1	0.5	3.9	0.6
Deployment	0.0	0.2	0.5	0.8

Figure 2. Obtained results

The estimation results are not very different from the previous with all “Nominal” Cost Drivers. Probably some parameters compensate other ones and the estimation result is very similar.

3.3. Cost Drivers and Scale Drivers

3.3.1. Software Cost Drivers

Cost drivers are used to capture characteristics of the software development that affect the effort to complete the project. A cost driver is a model factor that "drives" the cost (in this case Person-Months) estimated by the model.

All COCOMO II cost drivers have qualitative rating levels that express the impact of the driver on development effort. These ratings can range from Extra Low to Extra High. Each rating level of every multiplicative cost driver has a value, called an effort multiplier (EM), associated with it. This scheme translates a cost driver's qualitative rating into a quantitative one for use in the model.

- The EM value assigned to a multiplicative cost driver's nominal rating is 1.00.
- If a multiplicative cost driver's rating level causes more software development effort, then its corresponding EM is above 1.0.
- If the rating level reduces the effort then the corresponding EM is less than 1.0.

The Software Cost Drivers are divided into:

Product Factors

- **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.
- **Data Base Size (DATA)** - This cost driver attempts to capture the effect large test data requirements have on product development.
- **Product Complexity (CPLX)** - Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations.
- **Developed for Reusability (RUSE)** - This cost driver accounts for the additional effort needed to construct components intended for reuse on current or future projects.
- **Documentation Match to Life-Cycle Needs (DOCU)** - Several software cost models have a cost driver for the level of required documentation.

Platform Factors

- **Execution Time Constraint (TIME)** - This is a measure of the execution time constraint imposed upon a software system.
- **Main Storage Constraint (STOR)** - This rating represents the degree of main storage constraint imposed on a software system or subsystem.

- **Platform Volatility (PVOL)** - “Platform” is used here to mean the complex of hardware and software (OS, DBMS, etc.); the software product calls on to perform its tasks.

Personnel Factors

- **Analyst Capability (ACAP)** - Analysts are personnel who work on requirements, high-level design and detailed design.
- **Programmer Capability (PCAP)** - The increasing role of complex COTS packages, and the significant productivity leverage associated with programmers’ ability to deal with these COTS packages, indicates a trend toward higher importance of programmer capability as well.
- **Personnel Continuity (PCON)** - The rating scale for PCON is in terms of the project’s annual personnel turnover: from 3%, very high continuity, to 48%, very low continuity.
- **Applications Experience (APEX)** - The rating for this cost driver (formerly labeled AEXP) is dependent on the level of applications experience of the project team developing the software system or subsystem.
- **Platform Experience (PLEX)** - The Post-Architecture model broadens the productivity influence of platform experience, PLEX, by recognizing the importance of understanding the use of more powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities.
- **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.

Project Factors

- **Use of Software Tools (TOOL)** - The tool rating ranges from simple edit and code, very low, to integrated life-cycle management tools, very high.
- **Multisite Development (SITE)** - Given the increasing frequency of multisite developments, and indications that multisite development effects are significant, the SITE cost driver has been added in COCOMO II.
- **Required Development Schedule (SCED)** - This rating measures the schedule constraint imposed on the project team developing the software.

3.3.2. Software Scale Drivers

The exponent E , that is used in the effort equation, is an aggregation of five *scale drivers* that account for the relative economies or diseconomies of scale encountered for software projects of different sizes.

- If $E < 1.0$, the project exhibits economies of scale. If the product's size is doubled, the project effort is less than doubled. The project's productivity increases as the product size is increased.
- If $E = 1.0$, the economies and diseconomies of scale are in balance. This linear model is often used for cost estimation of small projects.
- If $E > 1.0$, the project exhibits diseconomies of scale.

The Scale Drivers are divided into:

- **Precedentedness (PREC)** - If a product is similar to several previously developed projects, then the precededentedness is high.
- **Development Flexibility (FLEX)** - The PREC and FLEX scale factors are largely intrinsic to a project and uncontrollable.
- **Architecture / Risk Resolution (RESL)** - This factor combines two of the scale factors in Ada COCOMO, "Design Thoroughness by Product Design Review (PDR)" and "Risk Elimination by PDR".
- **Team Cohesion (TEAM)** - The Team Cohesion scale factor accounts for the sources of project turbulence and entropy because of difficulties in synchronizing the project's stakeholders: users, customers, developers, maintainers, interfacers, others.
- **Process Maturity (PMAT)** - The procedure for determining PMAT is organized around the Software Engineering Institute's Capability Maturity Model (CMM).

4. Identifying the tasks and their schedule

1. RASD (Requirement Analysis Specification Document)

- Start Date 15/10/2015
- End Date 6/11/2015

2. DD (Design Document)

- Start Date 12/11/2015
- End Date 4/12/2015

3. Code Inspection Document

- Start Date 9/12/2015
- End Date 05/01/2016

4. ITPD (Integration Test Planning Document)

- Start Date 07/01/2016
- End Date 21/01/2016

5. Project Planning Document

- Start Date 22/01/2016
- End Date 02/02/2016

5. Allocation of members to tasks

1. RASD (Requirement Analysis Specification Document)

- Working hours per member: 40 hours each
- Tasks:
 - Marija Mavcheva: Introduction, Alloy Modeling, Scenarios, Class Diagram, State Chart Diagrams
 - Andrijana Mirchevska: Overall Description, Requirements, Use Case Diagrams, Sequence Diagrams.

2. DD (Design Document)

- Working hours per member: 30 hours each
- Tasks:
 - Marija Mavcheva: Requirement traceability, Architectural Design, Component Interface, Algorithm Design
 - Andrijana Mirchevska: Introduction, Architectural Design, Selected architectural styles and patterns, Algorithm Design

3. Code Inspection Document

- Working hours per member: 56 hours each
- Tasks:
 - Marija Mavcheva: Functional role of assigned class, Method 1: createNamingContext()
 - Andrijana Mirchevska: Assigned class, Method 2: addEnvironment

4. ITPD (Integration Test Planning Document)

- Working hours per member: 27 hours each
- Tasks:
 - Marija Mavcheva: Integration Strategy, Tools and Test Equipment Required.
 - Andrijana Mirchevska: Individual Steps and Test Description

5. Project Planning Document

- Working hours per member: 10 hours each
- Tasks:
 - Marija Mavcheva: COCOMO Approach
 - Andrijana Mirchevska: Functional Points

The total hours of work spent during all phases of the project are 326 hours.

$$326 \text{ hours} / (35 * 4) \text{ hours} = 2.33 \text{ Person/Months};$$

We suppose that a man can work 35 hours in a week so $35 * 4$ is a number of hours that man works in a month.

6. Risks and recovery actions

A risk is any anticipated unfavorable event or circumstances that occur while the project is underway.

In order to avoid risks as much as possible we will follow a risk management: reducing the impact of all kinds of risks that might affect a project.

Main categories of risks:

- *Project risks*

In this category of risks we are considering the following factors:

- **Budgetary** – The budget is not estimated good, for the requirements of the project to be done successfully;
- **Schedule** – The final date calculated for ending the project is not respected (not respecting deadlines);
- **Personnel** – The members of the team can't meet regularly at least once a week;
- **Customer-Related Problems** – There are some serious problems, such as not completing the need and wishes the customer had in the beginning of the project;
- **User Inconvenience** – The interface of the app might not be user friendly if it is complex for using and has unclear design.

- *Technical risks*

- Potential design, implementation, interfacing, testing, and maintenance problems.
- E.g. incomplete specification, changing specification, etc.

- ***Business risks***

- An excellent product that no one wants, losing budgetary, etc.

Strategies used for risk containment:

- ***Avoid the risk***

- Discuss with customer to reduce the scope of the work
 - Giving incentives to engineers to avoid the risk of manpower turnover, etc.

- ***Risk reduction***

- Planning ways to control the damage due to a risk

If there is risk that some key personnel might leave, new recruitment may be planned

- **Shared product vision**

- Sharing product vision based upon common purpose, shared ownership, and collective commitment
 - Focusing on results

- **Teamwork**

- Working cooperatively to achieve a common goal
 - Pooling talent, skills, and knowledge

- **Forward-looking view**

- Making surveys to collect information about user experience
 - Thinking toward tomorrow, identifying uncertainties, anticipating potential outcomes
 - Managing project resources and activities while anticipating uncertainties

7. References

- Slides for Cost Estimation, COCOMO II and Function Points, (Beep platform)
- Assignments Rules and Group Registration document (Beep platform)
- RASD, DD
- COCOMO II – Model Definition Manual, Version 2.1, 1995 – 2000 Center for Software Engineering, USC
- Software Risk Management by Gunjan Patel
- Online tool for COCOMO II: <http://csse.usc.edu/tools/COCOMOII.php>