Politecnico di Milano

2015-2016

Software Engineering 2: “MyTaxiService”
Project Plan

version 1.0

Author: Nenad Petrovic

•

Contents

•

Function point analysis

In this chapter, function point approach is going to be used in order to estimate the project size.

This technique is used for evaluation of project dimension based on functionalities of the application that is going to be developed. The functionalities list is derived using the previously completed RASD document.

Since it is common for computer systems to interact with other computer systems, a boundary must be drawn around each system to be measured prior to classifying components. This boundary must be drawn according to the user's point of view. In short, the boundary indicates the border between the project or application being measured and the external applications or user domain. Once the border has been established, components can be classified, ranked and tallied.

For each of them, classification is performed to the corresponding category and complexity value is assigned in order to perform the necessary calculation.

Classification and counting

According to the form of this approach, the functionalities have been grouped in:

- Internal Logical Files (ILF's) - a user identifiable group of logically related data that resides entirely within the applications boundary and is maintained through external inputs.

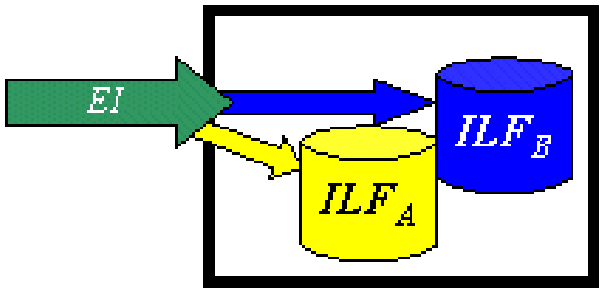
  Here belongs data that needs to be saved, used and managed inside the application itself. In this case (MyTaxiService), we need to store information about: users, taxi drivers, administrators, reports and drive events.

- External Interface Files (EIF's) - a user identifiable group of logically related data that is used for reference purposes only. The data resides entirely outside the application and is maintained by another application. The external interface file is an internal logical file for another application.

Here belongs data coming outside the boundaries of MyTaxiService, but is necessary to be used in order to achieve the wanted functionalities – GPS coordinates data, time estimation data, distance estimation data, person code data, driving license data, car id data.

GPS coordinates data, time estimation and distance estimation data is obtained from external GPS services, while the rest (person code, driving license, car id) is obtained from city government database, so this data could be generally divided into two categories – GPS data and government data.

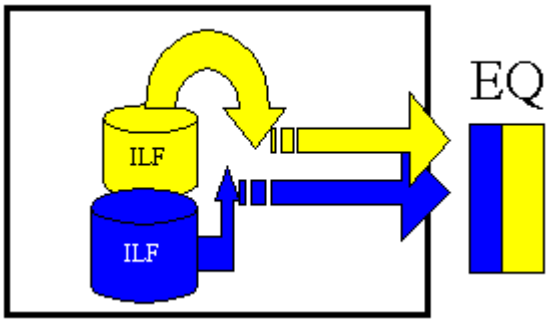
- External Inputs (EI) - is an elementary process in which data crosses the boundary from outside to inside. This data may come from a data input screen or another application. The data may be used to maintain one or more internal logical files. The data can be either control information or business information. If the data being considered is control information it does not have to update an internal logical file. The graphic represents a simple EI that updates 2 ILF's (FTR's).



Here input operations belong operations – log in, log out, register, profile modification, taxi request, delete users, update to taxi drivers, downgrade drivers, write report, request taxi, drive response request, drive offer response, manual taxi availability change.

- External Inquiry (EQ) - an elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files. The

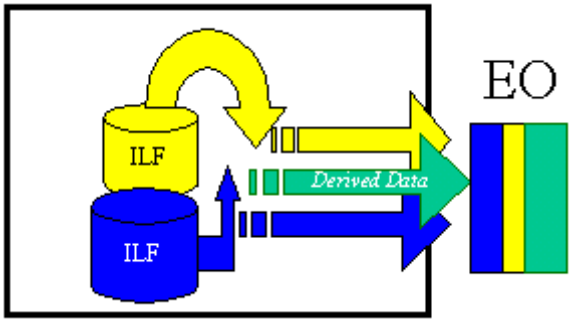
- input process does not update any Internal Logical Files, and the output side does not contain derived data. The graphic below represents an EQ with two ILF's and no derived data.



Here belong the following inquires: profile information visualization, view report, view drive, browse users, view user, visualize drive event, view offer/request summary.

.

• External Outputs (EO) - an elementary process in which derived data passes across the boundary from inside to outside. Additionally, an EO may update an ILF. The data creates reports or output files sent to other applications. These reports and files are created from one or more internal logical files and external interface file. The following graphic represents on EO with 2 FTR's there is derived information (green) that has been derived from the ILF's



•

Here belongs the taxi zone determination, cost estimation and taxi dispatching.

This data is available to developers using MyTaxiService API that is previously described.

Complexity and calculations

The following table outline the number of Functional Point based on funtionality and relative complexity:

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

Using the Function Points method, the estimation of the effort required to complete the application depends on the functionalities the application has to offer. Specifically, the number of FPs can be computed as the weighted sum of function types using the coefficient of the table above. In what follows, the calculation is performed step by step:

- Internal Logic Files (ILFs):

The application includes a number of ILFs that will be used to store the information about users, taxi drivers, administrators, reports and drive events.

- Each of these entities has medium structure complexity as it is composed of a medium number of fields except for administrators and reports, which have simple structure with small number of fields.

  This means that we will come out with 3 x 10 + 2 x 7= 30 + 14 = 44 FPs concerning ILFs.

- External Interface Files (EIFs):

  The application features two categories of EIFs – GPS data (coordinates, time, distance) and government data (person code, car id, driving license). The first is considered with medium complexity, while second is simple.

  The result is: 3 x 7 + 3 x 5 = 21 + 15 = 36 FPs

- External Inputs (EIs)

  The application interacts with the different types of users to allow them to log in, log out, register, profile modification, taxi request, delete users, update to taxi drivers, downgrade drivers, write report, drive request response, drive offer response, manual taxi availability change.

  - Login/logout: these are simple operations, so we can adopt the simple weight for them. 2 x 3 = 6 FPs
  - Register: this is a medium operation, so (involves check with external interface file) we can adopt the medium weight. 1 x 4 = 4 FPs
  - Profile modification: this is also a simple operation: 1 x 3 = 3 FPs
  - Taxi request: this is a complex operation because it includes many sub-operations 1 x 6 = 6 FPs
  - Offer/taxi request response: these are two medium complexity operations 2 x 4 = 8 FPs
  - Write report: simple operation 1 x 3 = 3 FPs
  - Update to taxi driver: complex operation, because involves data input, data check, and many entities are involved: 1 x 6 = 6 FPs

-
  - Downgrade taxi driver: medium operation 1 x 4 = 4 FPs
  - Manual taxi availability change: simple 1 x 3 = 3 FPs
  - Delete user: simple 1 x 3 = 3 FPs

    The sum is: 6 + 4 + 3 + 6 + 8 + 3 + 6 + 4 + 3 + 3 = 46 FPs

- External Output (EO)

  Here belongs the taxi zone determination, cost estimation and taxi dispatching. Each of these operations is quite complex, because it involves mathematical calculations (taxi zone determination, cost estimation) and usage of algorithms and data structures in order to create external output (taxi dispatching uses taxi queues and operations). So, here we will have the total number of function points:

  3 x 7 = 21 FPs

- External Inquiries (EQ)

  Here belong the following inquires: profile information visualization, view report, view drive, browse users, view user, visualize drive event (realtime), view offer/request summary.

  - Profile information visualization: simple 1 x 3 = 3 FPs

  - View report/drive: medium complexity, involves many entities 2 x 4 = 8 FPs

  - View user/browse users: simple 2 x 3 = 6 FPs

  - Visualize drive event (realtime): complex, involves usage of many enitites : 1 x 6 = 6 FPs

  - View offer/request summary: medium 2 x 4 = 8 FPs

  Sum: 3 + 8 + 6 + 6 + 8 = 31 FPs

- UFP calculation

  Finally, we sum previously calculated results : 44 + 36 + 46 + 21 + 31 = 178 FPs.

- This value can be used directly 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 for effort estimation.

Category	Simple	Medium	Complex	Sum [FPs]
ILF	- Admin - Drive event - Report	- User - Driver		44
EIF	- Fiscal codes - Car ID - Driving license	- Coordinates - Distance - Time		36
EI	- Login - Log out - Profile modification - Write report - Delete user - Availability change	- Downgrade driver - Register - Offer response - Request response	- Update to driver - Taxi request	46
EO			-taxi zone determination	21

•

			-cost estimation -dispatching	
EI	- profile info visualization - view user - browse users	- view report -view drive - view request summary - view response summary	-visualize drive event	31
Total				178

•

COCOMO approach

In this chapter, COCOMO (http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf) approach is used in order to estimate the effort and time needed to develop MyTaxiService. First, using results from previous FP calculations and statistics for project size, the size of the project is going to be estimated in source lines of code (SLOC). After that, effort is going to be calculated in number of persons/months. When effort calculation is completed, the result is used in order to estimate the project duration in months, and, finally, using previous two results (effort and duration) – the number of people needed to work on project is calculated.

The exponent E in equation is an aggregation of five *scale factors* (SF) that account for the relative economies or diseconomies of scale encountered for software projects of different sizes [Banker et al. 1994].

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. Some project economies of scale can be achieved via project-specific tools (e.g., simulations, testbeds), but in general these are difficult to achieve. For small projects, fixed start-up costs such as tool tailoring and setup of standards and administrative reports are often a source of economies of scale.

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. This is generally because of two main factors: growth of interpersonal communications overhead and growth of large-system integration overhead. Larger projects will have more personnel, and thus more interpersonal communications paths consuming overhead. Integrating a small product as part of a larger product requires not only the effort to develop the small product, but also the additional overhead effort to design, maintain, integrate, and test its interfaces with the remainder of the product. See [Banker et al. 1994] for a further discussion of software economies and diseconomies of scale.

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.

First, let' consider the case with nominal values, and after that the case with derived values from customized scale and cost drivers.

To pass from FP to SLOC we use an average conversion factor of 46 as described at http://www.qsm.com/resources/function-point-languages-table,

$$178\ FPs * 46 = 8188\ SLOC$$

A first estimation is based on FPs approach converted to SLOC. We Consider a project with all "Nominal" ost Drivers and Scale Drivers would have an EAF of 1.00 and exponent E of 1.0997. Following this formula

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

we obtain $effort = 2.94*(1.0)*(8.188)^{1.0997} = 29.687$ *Person-Months*

EAF: Effort Adjustment Factor derived from Cost Drivers.
E:Exponent derived from Scale Drivers.
Now we calculate the schedule (duration) of project in month with the following formula

$$Duration = 3.67*(effort)^{E}$$

We consider an exponent E of 0.3179

$$Duration = 3.67*(29.687)^{0.3179} = 10.784\ Months$$

Now we can estimate the number of people needed to complete the project with the following formula

$$N_{people} = effort/Duration$$

•

$$N_{people} = 29.687/10.784 = 3\ persons$$

Almost 30 person-months looks like a very big value for this project. This is a result of nominal value which are rough approximation and can't be taken as precise estimation.

So, in what follows, more precise calculation is going to be performed, using scale and cost drivers that take values specific to this project and real situation.

Scale drivers

These values are evaluated according to the following table:

Table 10. Scale Factor Values, SF_j, for COCOMO II Models

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

- Precedentedness:
  It reflects the team's previous experience with this kind of projects. Since for this team it was the first experience using this framework and these development methodologies, such as J2EE and Android, this value will be low.

- Development flexibility:
  It reflects the degree of flexibility in the development process. The teachers and teaching assistants constructed the assignments giving the general specifications without going too much in details, making this project development very flexible, so, for this reason this value is going to be set to high – general conformity.

- Risk resolution:
  Reflects the extent of risk analysis carried out. This value will be high ,considering this project.

- Team cohesion:
  Reflects how well the development team know each other and work together. Let's assume that this is team's first project and that they didn't know each other previously. Also, let's assume that there are synchronization problems, such as different academic assignments during project for each of the members. So, this driver will be high – largely cooperative.

- Process maturity:
  This was evaluated around the 18 Key Process Area (KPAs) in the SEI Capability Model.

  There are five levels of process maturity, level 1 (lowest half) to level 5 (highest). The CMM specifies "what" should be in the software process rather than "when" or "for how long".

  The CMM level 1 is for organizations that don‟t focus on processes or documenting lessons learned. The CMM level 1 is for organizations that have implemented most of the requirements that would satisfy CMM level 2. In CMM‟s published definition, level 1 (lower half) and (Upper half) are grouped into level 1.

- To be rated at a particular level, the organization should demonstrates capabilities in a set of Key Process Areas associated with a specific CMM level. The capabilities demonstrated in moving from lower levels to higher levels are cumulative. For example, level 3 organizations should show compliance with all key process areas in levels 2 and 3. The CMM process maturity framework is presented in Table.

  Because of the goals were consistently achieved these values will be set to high, level 3.

Process maturity Description	CMM Level 1 (lower)	CMM Level 1 (upper)	CMM Level 2	CMM Level 3	CMM Level 4	CMM Level 5
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Values	7.80	6.24	4.68	3.12	1.56	0.00

Using the formula below to calculate the exponent factor value from scale drivers:

$$E= B + 0.01\times \sum SFj,$$

where we have constant B=0.91.

The results are resumed in the following table:

Scale Driver	Factor	*Value*
Precedentedness	Low	4.96
Development Flexibility	High	2.03

Risk Resolution	High	2.83
Team Cohesion	High	2.19
Process Maturity	High	3.12
Total:		15.13
$E = B + 0.01 \times \sum SFj$		1.0613

•

Using the previously obtained Total(SD) from the table, the factor E (exponent) is calculated:

E=0.91+ 0.01 x Total(SD)=0.91 + 0.01 x 15.13 = 1.0613

Cost drivers

In this chapter, cost drivers are going to be analyzed in order to determine the EAF. The Effort Adjustment Factor in the effort equation is simply the product of the effort multipliers corresponding to each of the cost drivers for our project.

- Required Software Reliability:

  According to RASD and the nature of this project itself, the reliability parameter is important (deals with real-world traffic, taxi driver profit, passengers and taxi drivers). Software failures don't have critical consequences in most cases, and the that would risk human life, so this could be set to high – due to possible loss of users (not satisfied with service) and taxi drive opportunities (financial loss).

Table 17. RELY Cost Driver

RELY Descriptors:	slight inconven-ience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.82	0.92	1.00	1.10	1.26	n/a

- Data Base Size:

  It translates the effects that large data have in our application.

  Let’s assume that test database size is equal to 512.0 KB and the program size is equal to 8188 SLOC, the division:

  D/P = 15.99 and then this parameter has a nominal value, since it is between 10 and 100 (according to the table below).

Table 18. DATA Cost Driver

DATA* Descriptors		Testing DB bytes/Pgm SLOC < 10	10 ≤ D/P < 100	100 ≤ D/P < 1000	D/P ≥ 1000	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.90	1.00	1.14	1.28	n/a

- Product Complexity:
  Set to high according to the new COCOMO II CPLEX rating scale.

Table 20. CPLX Cost Driver

Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.73	0.87	1.00	1.17	1.34	1.74

- Required Reusability:

Table 21. RUSE Cost Driver

RUSE Descriptors:		none	across project	across program	across product line	across multiple product lines
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.95	1.00	1.07	1.15	1.24

In our project there are different reusable components since the aim was to design the system as modular as possible (common functionalities for all types of users). Also, it should be taken into account that for API development high reuse of previously developed components is going to be performed, so this parameter is therefore set to high.

- Documentation match to life-cycle needs:
  This parameter describes the relation between the provided documentation and the application requirements.

  In our system to be described has been expressed in the RASD and in the DD.

  On the other hand, it should be taken into account that code inspection was assignment not really related to this project and unit testing hasn't been performed. Also, the important thing is that, no implementation is performed and many lifecycle phases are not covered by the documentation.

- But, for this calculation, it is assumed that implementation is going to be performed, so this value is low, because not all phases are covered entirely.

Table 22. DOCU Cost Driver

DOCU Descriptors:	Many life-cycle needs uncovered	Some life-cycle needs uncovered.	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.81	0.91	1.00	1.11	1.23	n/a

- Execution Time Constraint:

As stated in RASD, there are many assumptions, limitations and constraints, so thi is set to high.

Table 23. TIME Cost Driver

TIME Descriptors:			≤ 50% use of available execution time	70% use of available execution time	85% use of available execution time	95% use of available execution time
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.11	1.29	1.63

- Main Storage Constraint: This parameter represents the degree of main storage constraint. In our application this parameter is not relevant (because the application doesn't need much storage), so this parameter won't be taken into account.

Table 24. STOR Cost Driver

STOR Descriptors:			≤ 50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.05	1.17	1.46

- Platform Volatility:

Table 25. PVOL Cost Driver

PVOL Descriptors:		Major change every 12 mo.; Minor change every 1 mo.	Major: 6 mo.; Minor: 2 wk.	Major: 2 mo.;Minor: 1 wk.	Major: 2 wk.;Minor: 2 days	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.87	1.00	1.15	1.30	n/a

- In our application is reasonable to consider as platforms the DBMS, the operating system, the browser, and the hardware as far as the environment concerns and compiler. The platform shouldn’t change, but updates are expected.

  But, considering the mobile counterpart of the application, both operating system and environment minor changes are expected (Android updates, Android Studio instead of Eclipse integration, etc.) so, the overall result is nominal, because the expected changes are natural.

- Analyst capability

A lot of effort was put in problem analysis (since, in reality, the complete assignment consists of documentation only, not of implementation) - analysis of the problem requirements, design and its potential integration if it had been developed, even the model correctness proof was performed using Aloy language.

Taking all these facts into account, the value should be set to very high.

Table 26. ACAP Cost Driver

ACAP Descriptors:	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.42	1.19	1.00	0.85	0.71	n/a

- Programmer Capability: This value is set to high and estimated as an overall result in cooperation and individual programming skill.

Table 27. PCAP Cost Driver

PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.34	1.15	1.00	0.88	0.76	n/a

- Application Experience:

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

Since the members have already successfully completed many projects in Java, have constructed many complete web sites and have more than 10 database-oriented applications in portfolio during past 3 years.

Our average knowledge about platforms as: databases, user interfaces and server side development are around 3 years (Databases, Databases 2, Advanced Databases, DBMS, Human-Computer interaction, Web programming, courses), so this parameter is set as high.

-

Table 29. APEX Cost Driver

APEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 years	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.22	1.10	1.00	0.88	0.81	n/a

- Platform Experience:

  This is set to high, because we haven't done previously any project with combination of these tools (Java EE and Android studio), but our knowledge and experience is high enough to deal with them.

Table 30. PLEX Cost Driver

PLEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.19	1.09	1.00	0.91	0.85	n/a

- Language and Tool Experience:

This parameter reflects the experience that is a consequence (in particular of previous two), so this is set to high, due to experience with Java programming language and Eclipse development.

Table 31. LTEX Cost Driver

LTEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.20	1.09	1.00	0.91	0.84	

- Personnel continuity:

This parameter is relevant in particular since in the current case our available time is less than half a year (1/2 of October, November, December, January, beginning of February) . For this reason we set it to very low.

Table 28. PCON Cost Driver

PCON Descriptors:	48% / year	24% / year	12% / year	6% / year	3% / year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.29	1.12	1.00	0.90	0.81	

- Usage of Software Tools:

Eclipse and Android Studio were used to manage dependencies of our project as libraries and development kits and Git for the repository management.

The most appropriate value is high.

•

Table 32. TOOL Cost Driver

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

- Multisite development:

This parameter is projection of how the team members have handled the distribution of development over distance and multiple platforms.

The team members used mobile phones, e-mail, Skype and Facebook (chat, file exchange, messenger calls, video calls) for communication and file exchange, and GitHub for version and changes management.

Considering a wide variety of different highly interavctive technologies used, this value is set to extra high (interactive multimedia).

.

Table 33. SITE Cost Driver

SITE: Collocation Descriptors: SITE: Communications Descriptors:	Inter-national Some phone, mail	Multi-city and Multi-company Individual phone, FAX	Multi-city or Multi-company Narrow band email	Same city or metro. area Wideband electronic communicat ion.	Same building or complex Wideband elect. comm., occasional video conf.	Fully collocated Interactive multimedia
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.22	1.09	1.00	0.93	0.86	0.80

- Required development schedule:

It was aim to distributed well over the available development time, but regardless of this fact, the implementation required high efforts at the later phases.

The problem formulation was extended by some requirements that would improve usage of this system in real world, and some minor things were adapted later. For these reason this parameter should be set to high.

Table 34. SCED Cost Driver

SCED Descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multiplier	1.43	1.14	1.00	1.00	1.00	n/a

•

In what follows, a table overview of the selected values for cost drivers is provided.

Cost Driver	Factor	*Value*
Required Software Reliability	High	1.10
Data Base Size	High	1.00
Product Complexity	High	1.17
Required Reusability	High	1.07
Documentation match to life-cycle needs	Low	0.91
Execution Time Constraint	High	1.11
Main Storage Constraint	Very Low	n/a
Platform Volatility	Nominal	1.00
Analyst Capability	Very High	0.71
Programmer Capability	High	0.88
Application Experience	High	0.88
Platform Experience	High	0.91
Language and Tool Experience	High	0.91
Personnel continuity:	Very Low	1.29
Usage of Software Tools	High	0.90
Multisite development	Extra High	0.80
Required development schedule	High	1.00
Product (EAF):		0.59

Effort Equation

After getting the values for both E and EAF, this final equation gives us the effort estimation measured in Person-Months (PM)

$$Effort := A * EAF * KSLOC^{E}$$

Where:

A → 2.94 (for COCOMO.2000)
EAF → product of all the cost drivers, equal to : 0.59 ;
E → exponent derived from Scale Drivers. Is calculated as:
B + 0.01 * sum{i} SF[i] := B + 0.01 *15.13 = 0.91 + 0.1371 = 1.0613;
in which B is equal to: 0.91 for COCOMO.2000 .
KSLOC → estimated lines of code using the FP analysis: 8.188

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

$$Effort := 2.94 * 0.59 * 8{,}188^{1.0613} = 16.157 \textit{ Person-Months}.$$

In conclusion , we could say that this value slightly differs from value obtained using all the nominal values (29.687 *Person-Months*), and looks much closer to reality.

Such result was expected due to much more things taken into account, deriving the values that correspond to custom project.

Schedule estimation

As far as the schedule estimation we are going to use the following formula:

$$Duration := 3.67 * Effort^{F}$$

Where:

$$F := 0.28 + 0.2 * (E – B) = 0.28+ 0.2*(1.0613-0.91) = 0.28 + 0.31026$$

Follows then:

$$Duration := 3.67 * 16.157^{0.31026} = 8.7$$

The duration calculated here is not even close to the actual time we had to our disposal for this project, that could be estimated being around 3.5 months. There are then two possibilities:

- Some of the estimations we made are not appropriate for the actual problem we are analyzing
- In reality the implementation part was not performed

The number of people required should be 16.157/8.7 = 1.4 . It means that 2 persons are more than enough for this project (compared to 3 persons using nominal values previously) .

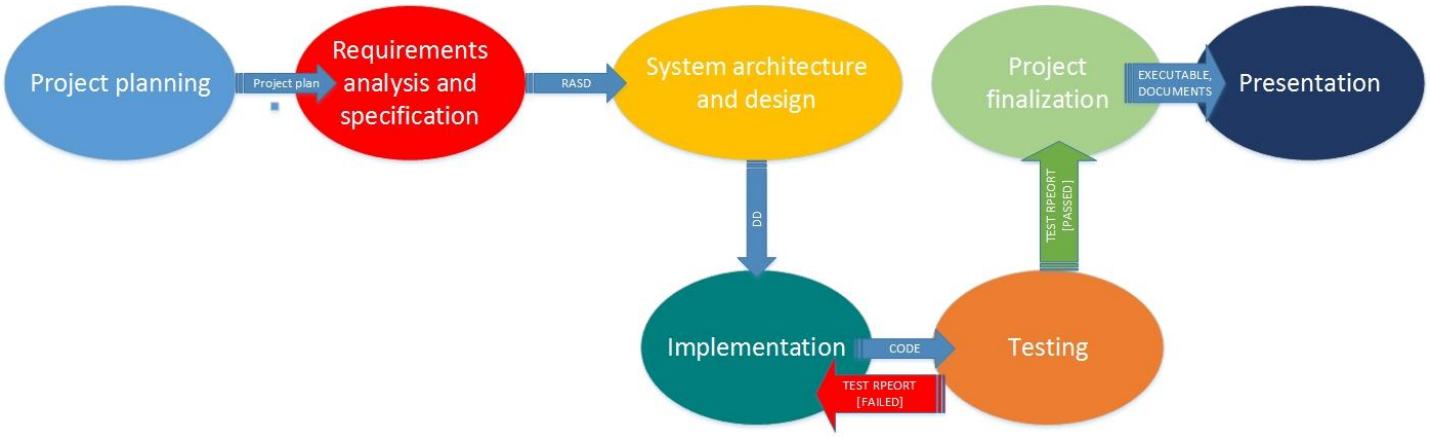
Scheduling and resource allocation

This chapter deals with project planning related to task scheduling and human resources allocation.

The project consists of 6 main parts, as seen below.

Project planning deals with scheduling, team engagement and resource allocation. After having this part done, project plan is completed and them is ready to start project.

Requirements analysis and specification results in RASD document which is, along Design document, the crucial part of the project. Implementation is parallel with System architecture and design, and uses decisions brought about architecture in design document, After that, the testing is performed. When some test fails, implementation is performed again (related to this part that was considered). Afterwards, if all tests passed, project finalization is done: executable application, manual and documents are delivered in definite versions and project could be presented to public or stakeholders.



The following terminology is going to be used:

Tasks are activities which must be completed to achieve the project goal. Phases are dividied in tasks, which shouldn't be long (maximum week or two).

Milestones are points in the schedule against which you can assess progress, for example, the handover of the system for testing.

Deliverables are work products that are delivered to the customer (for example, design document).

First, the project is going to be broken down into tasks. They are denoted as T1,T2…TN.

Each of the tasks has starting and ending date (duration). It could have milestones – M1,M2…Mn and deliverables denoted as D1,D2,…DN.

Dependencies stand for tasks or deliverables that must be completed before the considered task starts.

Last three columns are percentage of participation of each of the named members in given task. For example, 0 or empty field mean that members is not involve in this task, while 100 means that the member does it alone.

A few words about team members.

According to the calculations (COCOMO), more than 2 persons are needed. So, the team will consist of highly-experienced project manager (5 years in industry), Expert (more than a decade) and Junior, who is a fresh computer science graduate.

Nenad is highly-experienced member of the team and is project manager. He is also skilled in programming, software and graphical design, but not in mobile applcaition development. He is the one who is the most responsible for the deliverables.

Expert is member with highest experience (more than a decade in software industry) and is hired by Nenad for this project. This member is not involved into all phases of project, because his work is costly. But, he is crucial in requirements analysis, architecture/algorithm design phases, which are the the most important for future development. He is not involved in implementation phase, because his labor costs too much to spend money on this – there is a junior member that will do this.

Junior member is newcomer, so he is the least experienced and skilled, but has enough experience and skill in programming and mobile application development, so he will be the most responsible for mobile app development part. He will also actively participate in testing, but will participate less in architecture design-related tasks. His knowledge and experience are, though, enough for deriving use case diagrams from specifications and sequence diagrams. He will also help Nenad in implementation of the system. This project is opportunity for junior member to obtain new knowledge and skills also, participating in design process by some percentage and requirements analysis.

In what follows, table representation of this information related to MyTaxiService project is given using a table.

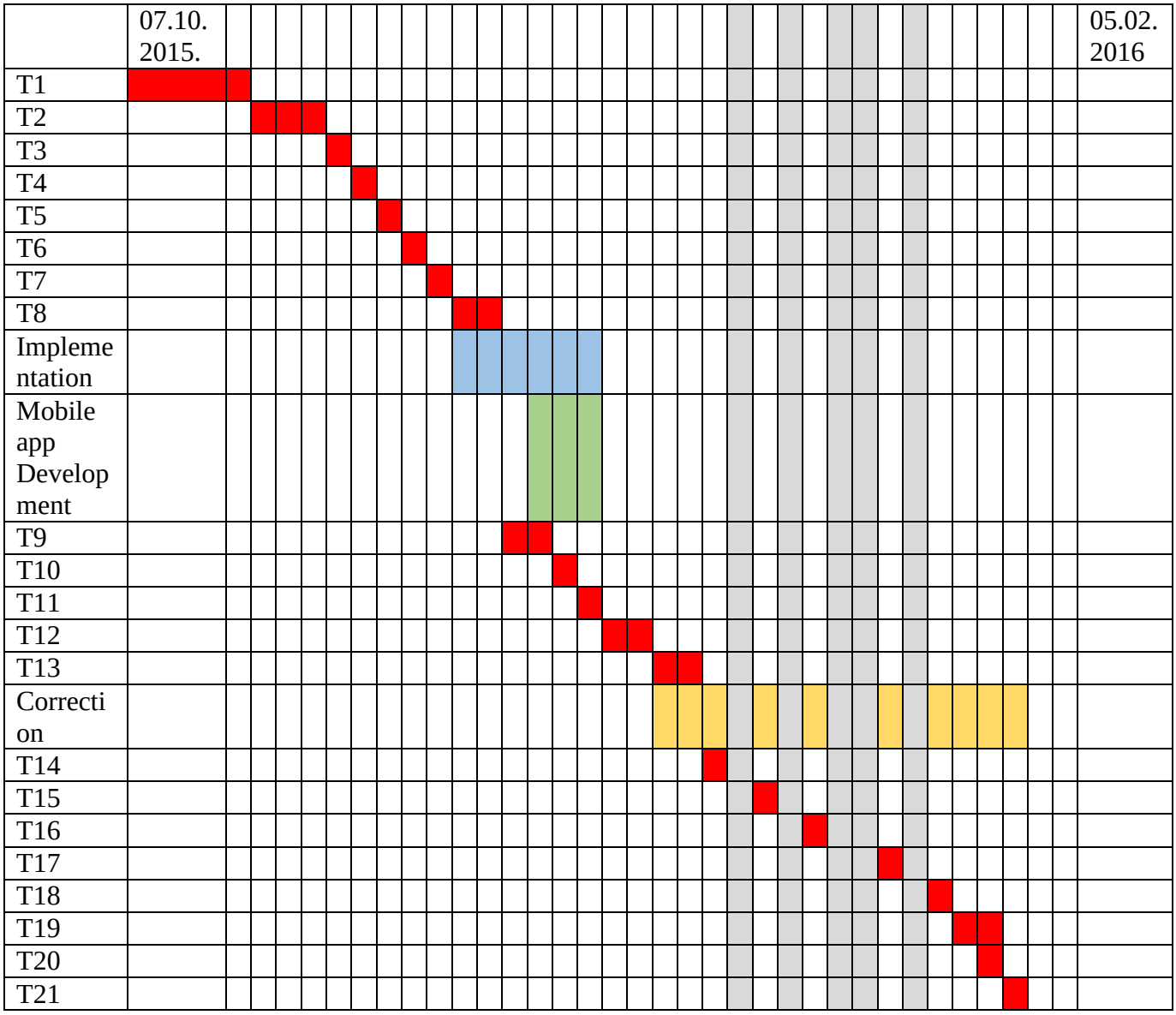
Task	Starting date	Ending date	Dependencies	Deliverables	Milestone	NE	EX	JU

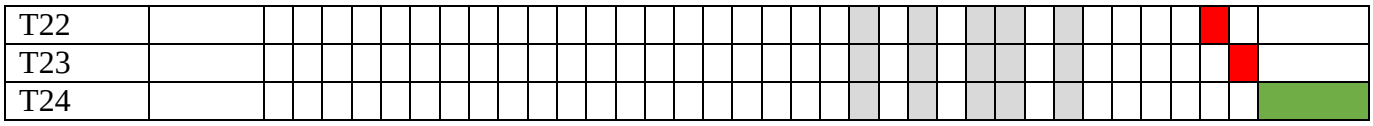
							NAD	PERT	NIOR
T1 – Team formation and planning			7-10-2015	14-10-2015		D1 – Project plan		100		
T2 – Requirements analysis			15-10-2015	27-10-2015			M1 – Goals, assumptions, constraint	25	50	25
T3 – GUI concept design			18-10-2015	21-10-2015	T2(M1)		M2 - Mockups, State transition diagram	50		50
T4 – Use cases			22-10-2015	25-10-2015	T3(M2)		M3 - Use case diagrams, Requirements level class diagram	30		70
T5- Scenarios identification and description			26-10-2015	29-10-2015	T4(M3)		M4 – Sequence diagrams	50		50
T6 – Model Verification			30-10-2015	3-11-2015	T4(M3), T5(M4)		M5 - Alloy model	50	25	25
T7 – RASD document completion			4-11-2015	06-11-2015	T2(M1),T3(M2),T4(M3), T5(M4)	D2 - RASD document		25		75
T8 – Architectural design	IMPLEMENTATION		07-11-2015	15-11-2015	T7 (D2)		M6 – High level components diagram	25/50/0	50/0/0	25/25/0
T9 – Interface specification			16-11-2015	23-11-2015	T7(D2), T8(M6)		M7 – Component, deployment and runtime view diagrams, sequence diagrams	50/50/0	25/0/0	25/50/0
T10 a– Algorithm design		MOBILE	24-11-2015	28-11-2015	T7(D2)		M8a – Algorithm diagrams	30/25/0	70/0/0	10/0/75
T10b – Databa			29-11-2015	30-11-2015	T7(D2)		M8b – Database schema	50/50/0		50/50/100

se design		A P P								
T11 – Design document completion			1-12-2015	4-12-2015	T8 (M6), T9(M7)	D3 – Design document D4 – application code		100/ 50/0		0/5 0/1 00
T12 – code inspection			5-12-2015	12-12-2015	IMPLEM ENTATI ON, MOBILE APP (D4)	D5- Code inspection document		50		50
T13 – unit test planning	C O D E C O R R E C T I O N		13-12-2015	20-12-2015	T7(D2), T11(D3)		M9 – test cases and specificati on	50/5 0		50/ 50
T14 – unit test			21-12-2015	23-12-2015	T13(M9)		M10 – unit test report	50/5 0		50/ 50
T15 – integrat ion test plannin g			26-12-2015	31-12-2015	T14		M11 – integratio n test cases and procedure	50/5 0		50/ 50
T16 – integrat ion test			2-1-2016	5-01-2016	T15(M11)		M12 – integratio n test report	50/5 0		50/ 50
T17 – System test plannin g			10-1-2016	12-01-2016	T16		M13 – system test specificati on and cases	50/5 0		50/ 50
T18 – System test			14-1-2016	18-01-2016	T17(M13)		M14 – system test report	50/5 0		50/ 50
T19 – Accept ance test plannin g			19-01- .2016	24-01-2016	T18		M15 – acceptanc e test specificati on	50/5 0		50/ 50
T20 - Accept ance test			25-01-2016	27-01-2016	T19(M15)		M16 – acceptanc e test report	50/5 0		50/ 50
T21 – Test docum ent prepara tion			28-01-2016	30-01-2016	T13-T20	D6 – Test document, D7 – Release version build		100/ 0		0/1 00

T22 – User manual preparation	31-01-2016	02-02-2016	T21(D7)	D8 – User manual		50	50
T23 – Presentation preparation	03-02-2016	05-02-2016		D9 – My Taxi Service presentation		50	50

In what follows, tasks are displayed against time as chart. Each slot (square) lasts 3.5 days in average. Red, blue, green and dark yellow are used to display when task is performed. Gray means that these days are holidays or non-working days: Catholic Christmas, New Year, Orthodox Christmas and Orthodox New Year. Names of tasks are given in enumeration form in order to keep clarity of the chart.





In what follows, another type of chart is going to be presented. It is resource allocation chart which shows who of the team members is allocated on which task during each time slot.

Nenad	T1	T1	T2	T2	T2	T3	T4	T5	T6	T7	T8	T8/impl	T9/impl	T10/impl	T11/impl	T12	T12
Expert			T2	T2	T2				T6		T8	T8	T9	T10/impl			
Junior			T2	T2	T2	T3	T4	T5	T6	T7	T8	T8/impl	T9/impl	T10/mobile	Impl/Mobile	T12	T12

Nenad	T13/corr	T13/corr	T14/corr	T15/corr	T16/corr	T17/corr	T18/corr	T19/corr	T19/corr	T20/corr	T21/corr	T22	T23
Expert													
Junior	T13/corr	T13/corr	T14/corr	T15/corr	T16/corr	T17/corr	T18/corr	T19/corr	T19/corr	T20/corr	T21/corr	T22	T23

Risk management

This chapter deals with risk management related to MyTaxiService project. It includes risk identification, classification and analysis, strategies to avoid risk and recovery plans in case that unwanted situation occurs.

Risk is a potential problem, which might happen or not. It involves change in mind, opinion, actions, places that could affect the project such way that unwanted losses or unwanted consequences could occur.

There are, in general, two types of strategies related to risk management.

Reactive risk strategies - nothing is done about risks until something really goes wrong.

Proactive risk strategies - steps for risk management are followed. Primary objective is to avoid risk and to have a contingency plan in place to handle unavoidable risks in a controlled and effective manner.

In this project, the second approach is going to be used, and it involves the following steps:

- Risk identification

  Identify all the possible risks that might occur.

- Risk analysis

  Analyze each risk to estimate the probability [L - low, M - moderate, H - high] that it will occur and the impact (i.e., damage) that it will do if it does occur.

- Ranking the risks by probability and impact

  Impact may be negligible, marginal, critical, and catastrophic

- Contingency plan development (to manage those risks with high probability and impact)

The following categories of risk are considered:

1. Project risks

   They threaten the project plan. If they become real, it is likely that the project schedule will slip and that costs will increase

2. Technical risks

They threaten the quality and timeliness of the software to be produced. If they become real, implementation may become difficult or impossible.

3. Business risks

They threaten the viability of the software to be built.

Sub-categories of Business risks:

- Market risk – building an excellent product or system that no one really wants.
- Strategic risk – building a product that no longer fits into the overall business strategy for the company.
- Sales risk – building a product that the sales force doesn't understand how to sell
- Management risk – losing the support of senior management due to a change in focus or a change in people.
- Budget risk – losing budgetary or personnel commitment

Considering the MyTaxiService, it could be concluded that project plan risks and technical risks could occur, but when it comes to business risks, not all the subcategories of risks are highly relevant.

Here, we can talk about management risk and budget risk, while market risks, strategic risk and sales risks are not so relevant here. MyTaxiService aims city government and users which will use this software for free (it is not a product for open market which has to compete with many similar applications), so there is no sales risk. City government already wants something like this, so there is no market nor strategic risk. Still, management and budget risks might occur.

After we have determined what risks exist for the project and assessed their importance, you need to choose a strategy for dealing with each risk if and when it comes into play.

In what follows, a table is given. The table consists of identified risks, estimation of their probability and how big are consequences in case of each risk. Also, for each risk is given a strategy which suggests how to deal with each of them.

Risk	Probability	Effects	Strategy
Requirements change	Moderate	Serious	Derive traceability information to assess requirements change impact; work on software flexibility.
Requirements have compliance issues (conflicts with law and legal regulations – as this software deals with personal data which needs to be authentic)	Low	Serious	Mention in the beginning what could be legal issues related to requirements and try to refromulate them in ordrer to

			avoid conflicts with law and legal regulations
Requirements fail to align with strategy - Requirements conflict with the firm's strategy (fair taxi management)	Moderate	Serious	Invovle stakeholders as much as possible during requirements analysis and formulation
Design lacks flexibility - A poor design makes change requests difficult and costly.	Moderate	Serious	Work on training and improving less experienced team members, but, in general, let only the most experienced members do this job, as design is critical part of the project and is heavily based on experience with similar projects.
Technology components aren't scalable - Components that can't be scaled to meet performance demands.	High	Serious	Consider different technologies (as alternatives) and start performance testing as soon as possible.
Technology components aren't compliant with standards and best practices - Non-standard components that violate best practices.	Moderate	Marginal	Use technology that is already has been approved as suitable for similar systems.
Project team lack authority to complete work	Low	Çatastrophic – could lead to project failure	Make clear in the beginning what is the authority needed (access to government databases, in this case) to complete the project and work on achieving it as early as possible.
Database performance	Moderate	Serious – could reject users	Consider using higher performance database or cloud.

Junior member not able to work on mobile app	Moderate	Catastrophic – mobile app development could fail completely, because he is the only one who is planned to be active for mobile app development and it relies mostly on him	Consider switching other team member to this task (either expert or project manager who is not so experienced in mobile apps development but is experienced and educated enough as an engineer to accept such challenge)