# MyTaxiService

## Project Reporting Document

Software Engineering 2

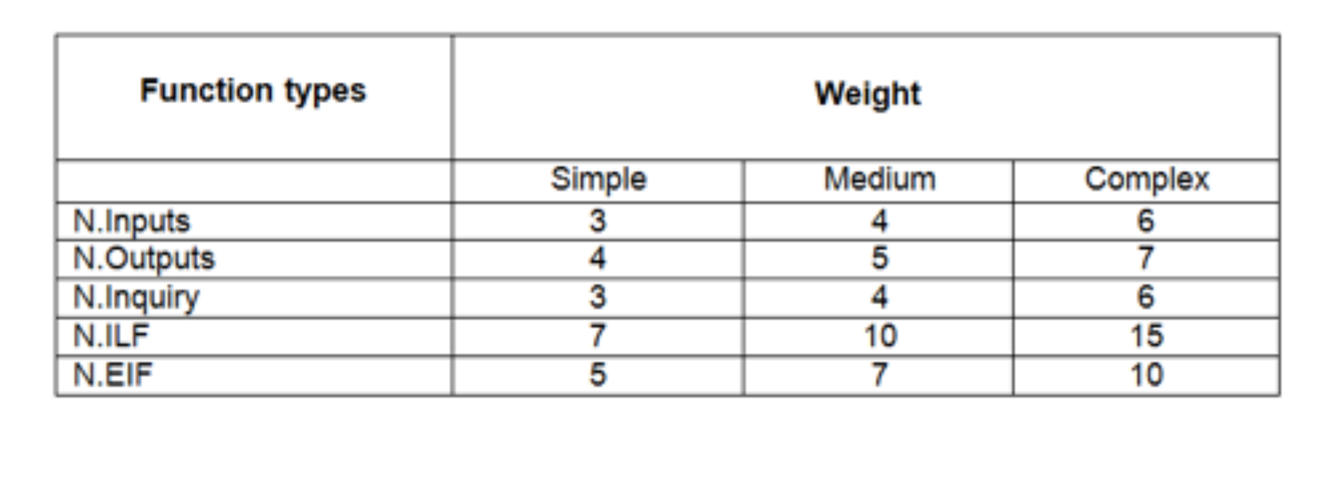
Authors:

Hasan Aliyev
Madat Mustafayev

Milan —2016
Keynote

# Contents

# 1. Function Points Approach



| Function types | Weight | | |
|---|---|---|---|
| | Simple | Medium | Complex |
| N.Inputs | 3 | 4 | 6 |
| N.Outputs | 4 | 5 | 7 |
| N.Inquiry | 3 | 4 | 6 |
| N.ILF | 7 | 10 | 15 |
| N.EIF | 5 | 7 | 10 |

Figure 1. Function types table

## 1.1 Internal Logical File ( ILF )

- Users
- Promotions
- Invoices
- Notifications

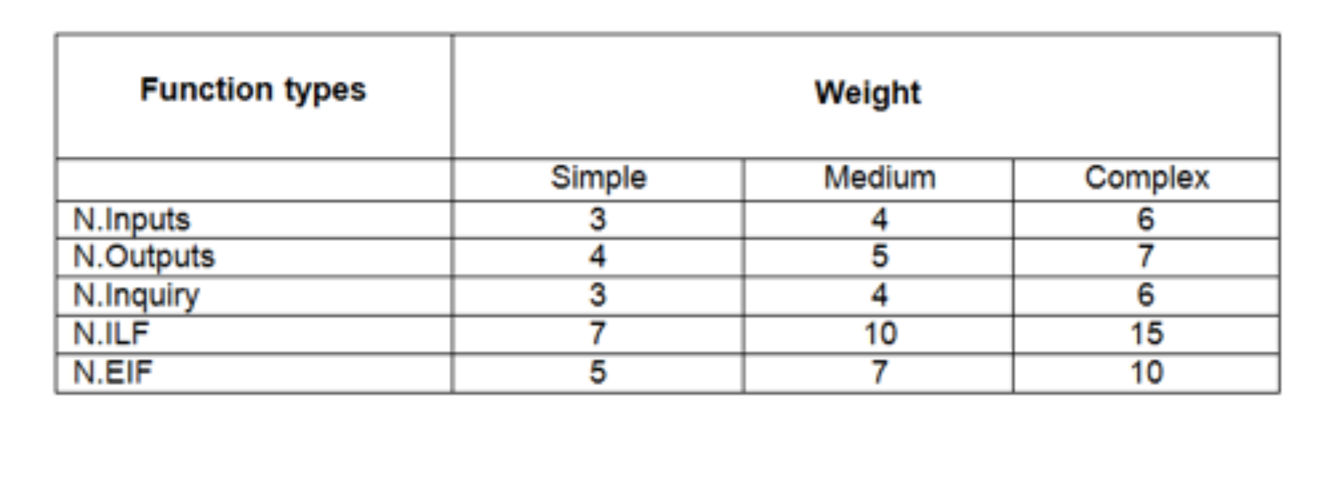
**There are four entities with simple structure. Thus we adopt a simple weight for all of them : 4x7 = 28 FPs.**

## 1.2 External Logical File ( EIF )

- Promotions
- Payment Transaction

**There are 2 entities and we adopt a medium weight for all of them: 2x7 = 14 FPs.**

# 1. Function Points Approach



| Function types | Weight | | |
|---|---|---|---|
| | Simple | Medium | Complex |
| N.Inputs | 3 | 4 | 6 |
| N.Outputs | 4 | 5 | 7 |
| N.Inquiry | 3 | 4 | 6 |
| N.ILF | 7 | 10 | 15 |
| N.EIF | 5 | 7 | 10 |

Figure 1. Function types table

## 1.3 External inputs ( Inputs )

- Order taxi -
- Set availability
- Set promo code
- Choose car class
- login/logout
- insert/delete information about promotion
- insert/delete information about new user

**There are 5 entities with simple structure (Order taxi,Set Availability,Set promo code,choose car class and login/logout). we adopt a simple weight for all of them : 5x7 = 35 Fps and also there are two complex structure with insert/delete information about promo code and also about user we assigned them complex structure 4x10=40 Fps**

# 1. Function Points Approach

### 1. 4 External Outputs ( Outputs )

- Creation invoices
- Fare estimation result

**There are 2 entities with complex structure. Thus we adopt a complex weight for all of them : 2x7 = 14 Fps**

### 1. 4 External Inquiry ( Inquiry )

- View profile
- The list of promotion
- Order History
- Operator can visualise information of customers

**There are 4 entities with medium structure. Thus we adopt a medium weight for all of them : 4x4 = 16 Fps**

In total we have 28+14+40+14+16=**<u>112</u>** Fps —> unadjusted function points. 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 to estimate the effort.

# 2. COCOMO Approach

## 2.0 COCOMO Approach

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, an updated version that adds J2EE of the table included in official manual http://sunset.usc.edu/research/COCOMOII/Docs/modelman.pdf).

112 FPs*46 = 5152 SLOC

**There are 2 entities with complex structure. Thus we adopt a complex weight for all of them : 2x7 = 14 Fps**

## 2. 1 COCOMO formula

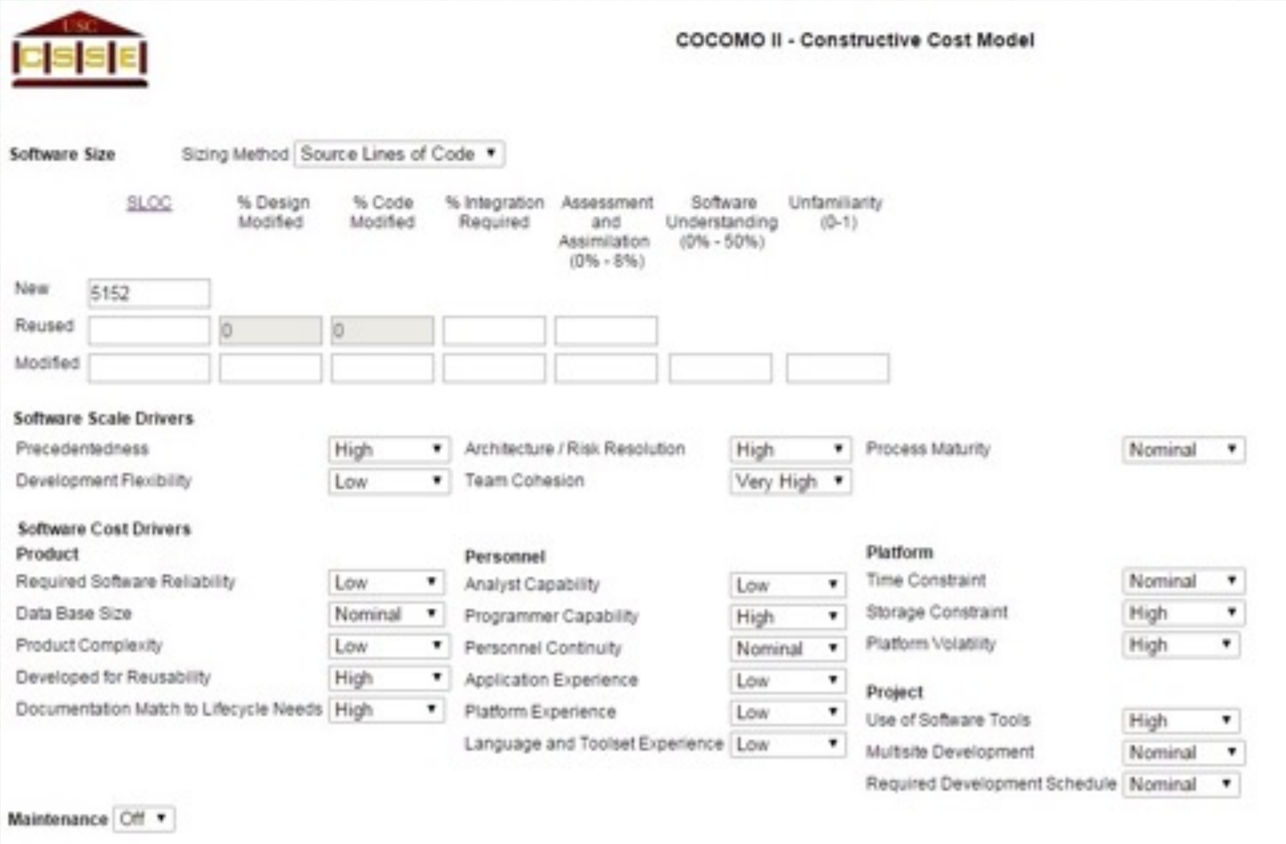
Effort =2:94 $(\text{KSLOC})_E$ *EAF

- EAF : The effort Adjustment Factor derived from the Cost Drivers.
- E : Is an exponent derived from five Scale Drivers
- KSLOC : 5,152

# 2. COCOMO Approach

## 2.2 COCOMO Result

These results are obtained with the following link: http://csse.usc.edu/tools/COCOMOII.php



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 | 5152 | | | | | | |
| Reused | | 0 | 0 | | | | |
| Modified | | | | | | | |

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 | High |
| 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 |

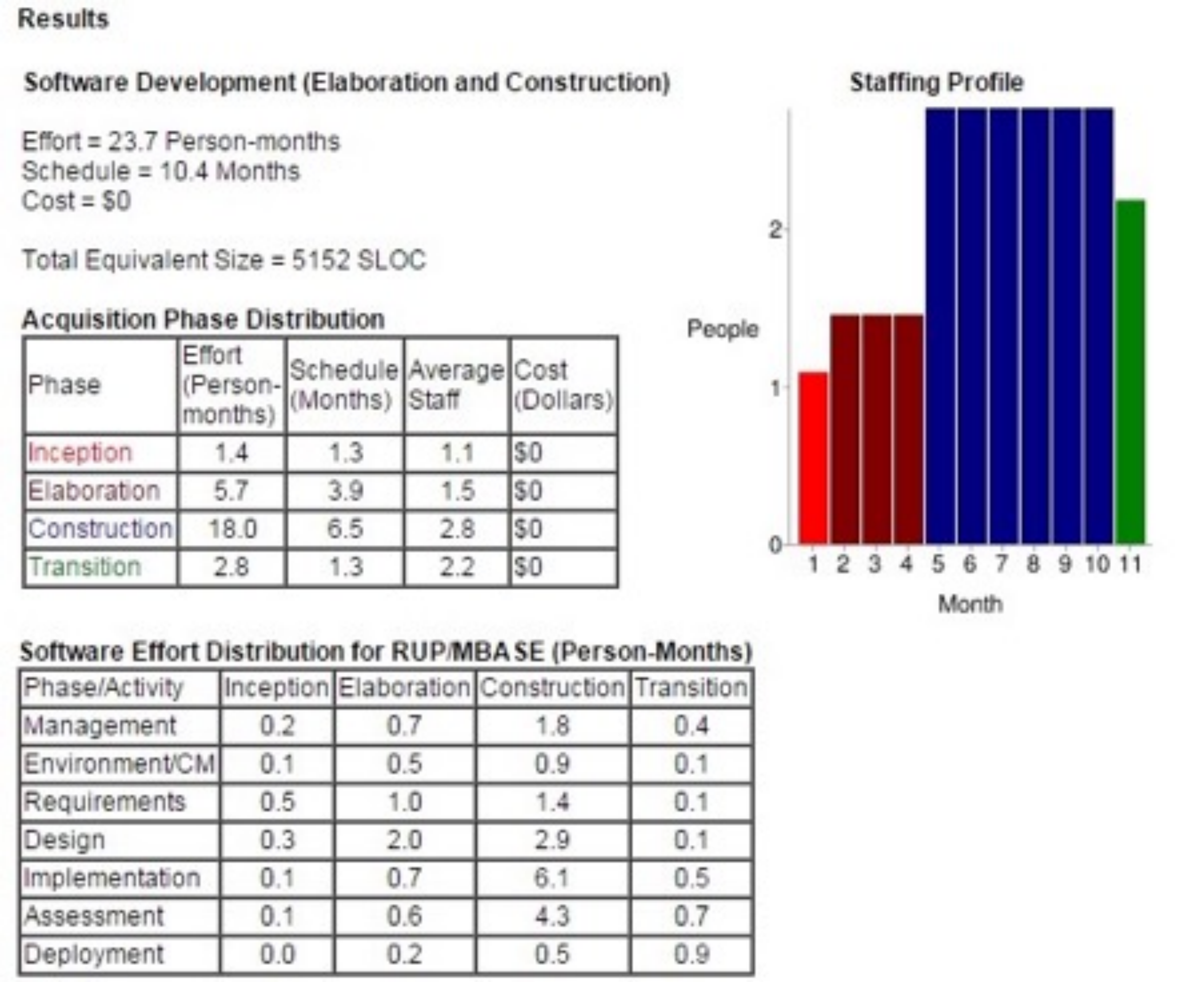
Maintenance: Off

Figure 2. Constructive Cost Model

# 2. COCOMO Approach

## 2.2 COCOMO Result

Number of people = Effort/Duration =23.7/10.4 = 2.27  people.To fulfil the effort required for this project 2 people are required, which  is slower higher with our number of team members.



Results

Software Development (Elaboration and Construction)

Effort = 23.7 Person-months
Schedule = 10.4 Months
Cost = $0

Total Equivalent Size = 5152 SLOC

Acquisition Phase Distribution

| Phase | Effort (Person-months) | Schedule (Months) | Average Staff | Cost (Dollars) |
|---|---|---|---|---|
| Inception | 1.4 | 1.3 | 1.1 | $0 |
| Elaboration | 5.7 | 3.9 | 1.5 | $0 |
| Construction | 18.0 | 6.5 | 2.8 | $0 |
| Transition | 2.8 | 1.3 | 2.2 | $0 |

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

| Phase/Activity | Inception | Elaboration | Construction | Transition |
|---|---|---|---|---|
| Management | 0.2 | 0.7 | 1.8 | 0.4 |
| Environment/CM | 0.1 | 0.5 | 0.9 | 0.1 |
| Requirements | 0.5 | 1.0 | 1.4 | 0.1 |
| Design | 0.3 | 2.0 | 2.9 | 0.1 |
| Implementation | 0.1 | 0.7 | 6.1 | 0.5 |
| Assessment | 0.1 | 0.6 | 4.3 | 0.7 |
| Deployment | 0.0 | 0.2 | 0.5 | 0.9 |

Figure 3. The result of  Constructive Cost Model

# 3. Tasks and Schedule

## 3.0 Tasks

Our main tasks for the mentioned project are consists of modules which is shown below:

| | |
|---|---|
| • **Task1:RASD** | **15.10.15-06.11.15** |
| • **Task2 Design Document** | **11.11.15-12-12-15** |
| • **Task3 Code Inspection** | **12.12.15-05.01.16** |
| • **Task4 Testing** | **07.01.16-22.01.16** |

## 3. 1 Schedule of tasks

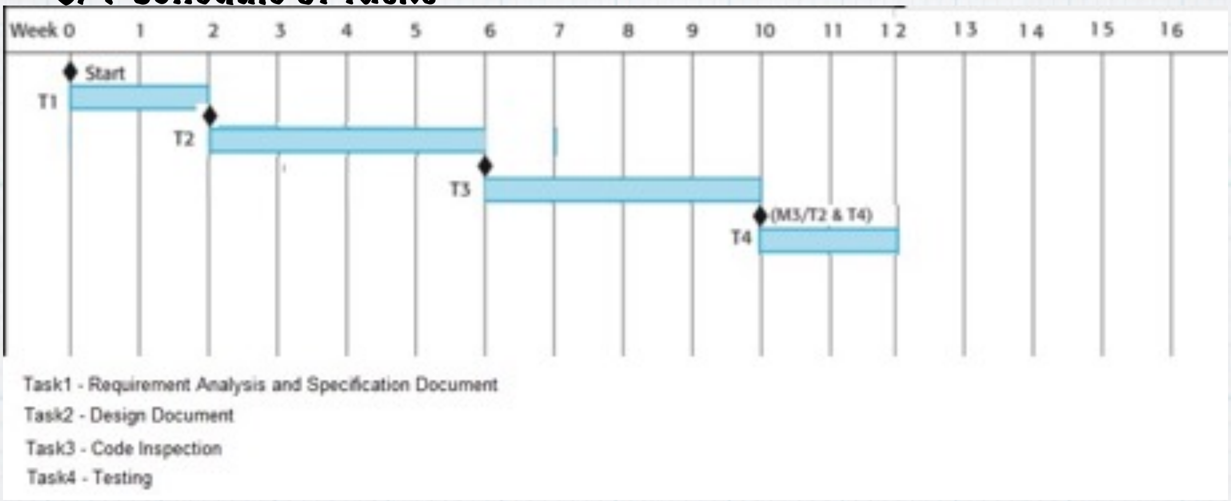


Figure 4. Activity bar chart

## 3. 1 Resource Allocation

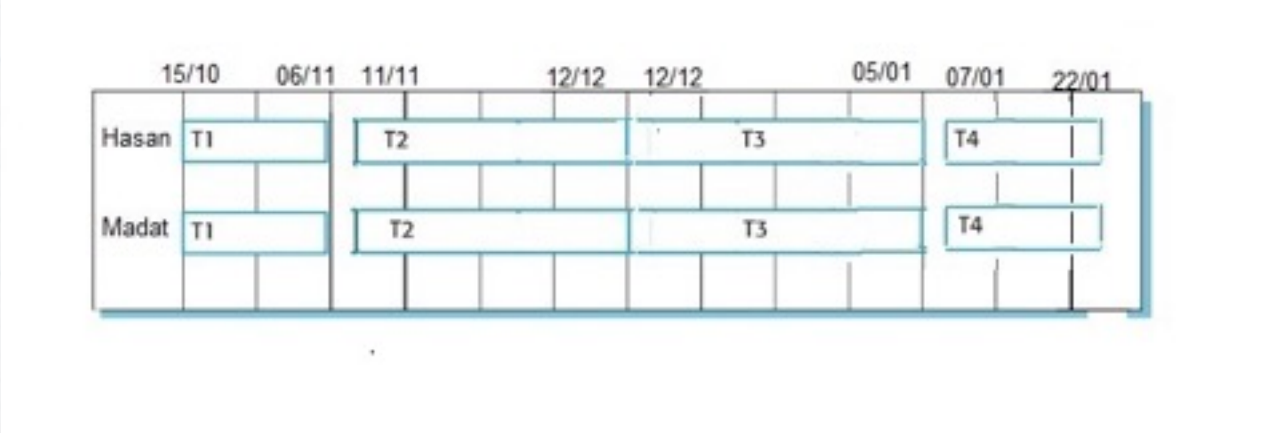


Figure 5 Staff allocation chart

# 4. Risks Management

## 4.0 Definition of the risks

A risk is a potential problem – it might happen and it might not.

**Conceptual definition of risk:**

- Risk concerns future happenings
- Risk involves change in mind, opinion, actions, places, etc.
- Risk involves choice and the uncertainty that choice entails

## 4.1 Steps for Risk Management

- Identify possible risks; recognize what can go wrong
- Analyze each risk to estimate the probability [L,M,H] that it will occur and the impact (i.e., damage) that it will do if it does occur
- Rank the risks by probability and impact
- Develop a contingency plan to manage those risks having high probability and high impact

In our project we could have some risks and below we highlight them.

| Risk | Probability | Effects |
|---|---|---|
| Personal shortfall | Moderate | Catastrophic |
| Unrealistic schedule | Lower | Catastrophic |

If above risks become real, it is likely that the project schedule will slip and that costs will increase. Despite the fact that, our team was consists of 2 people we did our best to finish our project on time.