TAXISERVICE

PROJECT PLAN

Authors: Jacopo Silvestri 773082 Simone Penati 850448
Reference Professor: Mirandola Raffaela
Release Date February 2016

POLITECNICO
DI MILANO

0. Index

1. Introduction

In this document we go through the project plan for our TaxiService software in two distinct subsequent approaches. First we will analyze our project complexity by evaluating its cost in terms of Function Points, by assigning a specific value to each feature included in our project. Secondly we will use the COCOMO approach to estimate the cost in terms of effort necessary to complete the creation of the Taxi Service software.

2. Function Point Approach

We will now proceed to analyze the complexity of our project. First we will assign at every feature our software will have to a FP Category and a complexity, in order to evaluate its value in terms of Function Points. In the following list there are the assignment we have performed and the FP table to which we will refer when calculating the total Function Points of our Project. All the functionalities have been derived from the R.A.S.D. and Design Document.

Functionality	Simple	Medium	Complex
External Input	*3*	*4*	*6*
External Output	*4*	*5*	*7*
I.L.F.	*7*	*10*	*15*
E.L.F.	*5*	*7*	*10*
External Inquiries	*3*	*4*	*6*

We identified the following Function Points, critical to the realization of TaxiService. Here we present them, divided by function type.

- **External Input (31 FPs):**
  - User Login/Log Out
    - These are simple operations to implement in any client-server system. Therefore we classify these features as simple and assign them a total of 2x3 = 6 FPs.
  - User Signup
    - As above this is considered a simple operation, which has only to create a new data in the main DBMS with all the information inserted by the User; we considered this to have simple weight, 1x3 = 3FPs.
  - Taxi Requests:
    - This particular functionality is at the core of our software and implies a direct communication between server and client and the creation of a new taxi queue. So we flag this functionality as medium, for its higher complexity. 1x4 = 4FP.
  - Taxi Reservations
    - As above, this is one of the main feature, but differently from the previous one we have a registration of the time and location at which the reservation is booked. Because of this additional functionality and the different behaviour with the system treat reservation, we flag this feature as complex: 1x6 = 6FP.

- Accept/Refuse call functionality
  - This is a simple operation that sends a simple notification to the system and then to the user. Simple weight: 1x3 = 3FP.
- Cancel Request/Reservation functionality:
  - This functionality erases a reservation/Request from the database. In virtue of its simplicity we assign a simple weight: 1x3 = 3FP.
- Update Status:
  - A simple operation that change a boolean value on the central Server. Simple Weight: 1x3 = 3FP.
- Update Personal Information:
  - Just as the SignUp Functionality this feature just modifies some of the user's data in the central DBMS. Analogously as the example before, we apply a simple weight: 1x3 = 3FP

- **External Output (17 FPs):**
  - Notifications
    - The notification system is at the center of our software: it has to manage all request, acceptances, refusals, reservations and so on. Because of this inherent complexity, we will assign a medium weight: 1x5 = 5FP.
  - Show Personal Information:
    - A simple output that allows the user to visualize its profile information. We have decided for simple weight: 1x4 = 4FP.
  - Show Taxi Code (for a Taxi Call):
    - A simple operation to show to any user that requested a taxi, the taxi's code that accepted his request. Simple weight 1x4 = 4 FP.
  - Show Taxi License Plate (for a Taxi Call):
    - Same as above. It is something that the user may want to visualize. Simple Weight: 1x4 = 4FP.

- **External Inquiries (9 FPs):**
  - Fetch Taxi Queue:
    - This functionality has to create a list of all the taxi while inquiring their data to fetch the information concerning availability e position. This is done, in order to collect all the fundamental data to execute the Dijkstra Algorithm and find the nearest available taxi to the User (Customer). Complex weight: 1x6 = 6FP.
  - Check Taxi Availability:
    - A simple inquiry that checks for the truthness of boolean value. Simple weight: 1x3 = 3 FP.

- **EIF (14 FPs):**
  - License Database:
    - Since the Licence DBMS is controlled by the government we assume a pessimistic approaches and say that the algorithm to access to a different DBMS with a different internal logic will be of a medium weight instead of simple: 1x7 = 7FP.
  - Taxi GPS Data:
    - As above, since the effort to chase a performing compatibility between our system and a COTS may very well be higher than expected, we have decided to assign to this feature a medium weight instead of a simple one: 1x7 = 7FP.

- **ILF (36 FPs):**
    - User Personal Information
        - Simple weight, as the profile system is common knowledge in every web services that requires an account. 1x7 = 7FP.
    - City Map:
        - Complex weight, as we have to create a dynamic map that will work alongside our taxi queue algorithm. 1x15 = 15FP.
    - Database of Taxis registered to the service:
        - Simple weight for the same reasons specified in the User Personal Informations. 1x7 = 7FP.
    - Taxi Queue:
        - Simple weight as the complexity of the creation of the taxi queue is already been allocated in its rightful feature above. 1x7 = 7FP.

Total Unadjusted Function Points (UFPs) = (36+14+9+17+31)UFPs = 107 FP.

2.1 COCOMO Approach

Now that we have an estimation for the complexity of our project, we will proceed to estimate the effort and duration of the project, using COCOMO.
The following list comprehends all the Cost Drivers and the value chosen. The KLOC estimate is in the order of ten (~10^4 Lines of Code).

Product Attributes

- Required Software Reliability
  - *very high - 1,40*
- Size Of Application Database
  - high - 1,08
- Complexity Of The Product
  - *nominal - 1,00*

Hardware Attributes

- Runtime Performance Constraints
  - *very high - 1,30*
- Memory Constraints
  - *nominal - 1,00*
- Volatility of the VM Environment
  - *low - 0,87*
- Required Turnabout Time
  - *nominal - 1,00*

Personnel Attributes

- Analyst Capability
  - *nominal - 1,00*
- Applications Experience
  - *nominal - 1,00*
- Software Engineer Capability
  - *high - 0,86*
- Virtual Machine Experience
  - *nominal - 1,00*
- Programming Language Experience
  - *high - 0,95*

Project Attributes

- Application of SE Methods
  - *high - 0,91*
- Use Of Software Tools
  - *high - 0,91*
- Required Development Schedules
  - *high - 1,04*

Estimating the project to be organic (a=3,2 ; b=1,05), with an EAF of 1,2, we obtain:
Effort = 3,2 * 1,2 *$(10)^{1,05}$ = 44 person-months.
Development Time = 2,5 * $44^{0,38}$ = 11 months.
People Required = 44/11 = 4 people.

3. Schedule and Resource Allocation

Give the available time and assuming that the actual project has started in October, we will base or schedule referring to our actual available time in that period. Using the amount of time we actually used to produce all the documentation concerning our project, and assuming the implementation part of the TaxiService software cannot begin until all the documentation is not finished (with the exception of the Integration Test Document), the time necessary to prepare our software's design can be approximate with 170-200 total hours of work, plus the time needed for the implementation. This value does not include the time needed for unit testing or integration, as those phases will be carried out during the implementation. We assume that we will already know all the useful knowledge, concerning the course of Software Engineering II, learned in this period of time.
Therefore we can write down the following schedule, assuming we will have roughly 30 hours of work per week (3 per day, plus approximately 15 per weekend). We also insert in the table an estimation of what can be the time spent to agrees on the term of our project and on all the requirement that it concerns with a problematic stakeholder:

Task	Worker	Hours Needed	Estimate Begin Date	Estimate Completion Date
R.A.S.D.	Penati, Silvestri	30 h	15st October 2015	23th October 2015
Discussing RASD with Stakeholder	Penati, Silvestri	15h	23th October 2015	26th October 2015
Design D.	Penati	30 h	26th October 2015	2th November 2015
Project Planning	Silvestri	30 h	26th October 2015	2th November 2015
Discussing DD, PP with Stakeholder	Penati, Silvestri	15h	2th November 2015	6th November 2015
Implementation*	Penati, Silvestri	22 month	6th November 2015	6th July 2017
Testing and Integration	Penati, Silvestri	60h + Testing	6th November 2015**	6th July 2017**

* Values based on our COCOMO effort estimation.

** We assume that the Testing and the Integration Phase is done while the implementation phase is still under its course. The hours of testing are included in the Implementation time estimate. The 60h is the time to create a complete reference document in order to use it to guide the Integration and Validation Phase of the project.

4. Risks Evaluation

We've decided to adopt a proactive strategy for our risk avoidance and management policy. Thus we will proceed to take in consideration any risk before it may affect the plan.

4.1 Risk Identification and Analysis

- Project Risks:
    - Problem with software's interactions between TaxiService and COTS components, in particular Identity Management and Statal Taxi Licence Database. (*Probability: Medium, Impact: Low*) 4.5
    - Problems generated from Stakeholder Discussions and requirement changes. (*Probability: Very Low, Impact: High*) 4
    - Drastic reduction of the hours of work available. (*Probability: Low, Impact: High*) 5

- Technical Risks:
    - Lack of knowledge in the Taxi IT Environment, in particular to respect with the complex interaction and integration with other subsystem. (*Probability: Low, Impact: Medium*) 4.25
    - Possible difficulties to implement the GPS feature. (*Probability: Low, Impact: Medium*) 4.25

- Business Risk:
    - Market Risks:
        - Since this software has been explicitly requested by our stakeholder, this project will not incur in any market risk.(*Probability: Very Low, Impact: High*) 4
    - Strategic Risks:
        - This risks should not subsist as we will have a direct and constant communication with our stakeholder to ensure that the product we are building will be exactly like he wants it to be. (*Probability: Very Low, Impact: Very High*) 4.75
    - Sales Risks:
        - As the market risk, the fact that TaxiService has been ordered and will be not placed on the market without a certain probability of sale, this risk does not subsist. (*Probability: Very Low, Impact: High*) 4
    - Management Risks:
        - Since our team is relatively small, the risk inherent to coordination and cooperation problematics are kept to a minimum. (*Probability: Very Low, Impact: Medium*) 3.25
    - Budget Risks:
        - The application may be too pricy to develop. (*Probability: Very Low, Impact: Low*) 2.5
        - We should not have any commitment problematics. (*Probability: Very Low, Impact: High*) 4

5.2 Risk Ranking by Probability and Impact

After identifying all the risks and analyzing their probability and impact, we decided to assign an integer value - from one to five - to both variables' severity, which now ranks from "*Very Low*" (=1) to "*Very High*" (=5). We then stated that the probability of our risks was of a higher concern for our project so while the probability contributes fully to the Risk Value, the impact contribution is multiplied by 0,75.

To break eventual ties, we gave the priority to Project Risks, then to Technical Risks, and lastly to Business Risks. Eventual further ties are broken considering situation per situation.

Rank	Risk	Risk Value
1	Revisions post-discussion	5
2	Strategic	4,75
3	Software/COTS Interaction	4,5
4	Knowledge lack	4,25
5	Difficulties implementing GPS	4,25
6	Reduction of work hours available	4
7	Budget	4
8	Market	4
9	Sales	4
10	Management	3,25

5.3 Develop Prevention and Contingency Plans

1. Problems generated from Stakeholder Discussions and requirement changes.
    - ➢ *Anticipate eventual problems by having clear ideas and making points since the preliminary meeting with the stakeholder, in order to avoid subsequent uncertainties or abrupt changes, and also any possible misunderstanding.*
    - ➢ Should the issue arise: *A good application design and architecture which increase the reusability and the flexibility of all our functions could help solving the problem of requirements' changes.*
2. Strategic Risks
    - ➢ *As for any problem emerging from discussions and meetings, we will maintain an open and constant communication with the stakeholder to ensure that the product fits in the company strategy.*
    - ➢ Should the issue arise: *Immediately organize a major meeting to re-discuss the terms of our agreement.*
3. Problem with software's interactions between TaxiService and COTS components
    - ➢ *We will get all possible support from the third party documentation and customer service to ensure that no compatibility issues arise between our proprietary system and the COTS components.*
    - ➢ Should the issue arise: *request the cooperation of the taxi company IT manager, to solve the problem as quickly as possible.*
4. Lack of knowledge in the Taxi IT Environment
    - ➢ *Get support from the stakeholder and possibly analyze any similar open source software to understand and reuse any valid and viable solutions.*
    - ➢ Should the issue arise: *Slow the development down, reallocating some of our time resources to study parts of the COTS and design a feasible solution.*
5. Possible difficulties to implement the GPS feature
    - ➢ *Analyze any similar open source software to understand and reuse any valid and viable solutions.*
    - ➢ Should the issue arise: *Seach for a COTS software that we could use to implement that feature for our system, since it is already widely in use in the Taxi IT Environment.*

6. Drastic reduction of the hours of work available
    - ➢ *We cannot prevent this issue to happen since it is beyond our possibilities: this issue could be caused by impending personal issues, health problems, or various other reasons that transcend our reach.*
    - ➢ Should the issue arise: *Re-organize our schedule to maximize our working efficiency.*
7. Budget Risks
    - ➢ *Using all the documentation we produced, we can be quite sure of the possible needed budget range.*
    - ➢ Should the issue arise: *Organize a meeting with the stakeholder to discuss the situation and the impact that huge budget reduction could have in the software requirements. Then, search for a feasible solution within the new given budget.*
8. Market Risks
    - ➢ *Since the application has been explicitly requested, there should not be any problem regarding its demand. We should however keep an open communication with the stakeholder to make sure every moment that the application we are building and he/she wants are one and the same.*
    - ➢ Should the issue arise: *Immediately contact another stakeholder in the same environment (i.e. other cities' taxi companies, car sharing or public transport agencies) to offer our product before the implementation phase ends.*
9. Sales Risks
    - ➢ *There should not be any uncertainty as to whether the product will sell or not, since it has been requested.*
    - ➢ Should the issue arise: *Same solution as above.*
10. Management Risks
    - ➢ *Keeping a constant and active communication within our group we can avert any management issue between us. The small size of the team also aids in this regard*.
    - ➢ Should the issue arise: *Immediately organize a small meeting to re-discuss the job schedule and the worker's' role.*