Politecnico di Milano

Master in Computer Science and Engineering

Software Engineering 2 Project: myTaxiService

PROJECT PLAN

Authors:
Filippo Leporati
Danilo Fusi

Professor:
Elisabetta Di Nitto

1 TABLE OF CONTENTS

2 INTRODUCTION

2.1 PURPOSE

The primary uses of this project plan are to document planning assumptions and decisions, facilitate communication among stakeholders, and document approved scope, cost, and schedule baselines.

2.2 SCOPE

myTaxiService application, which is an information service, aims at optimizing the taxi service of a large city.
In particular, it wants to:
Business Goal 01) Simplify the access of passengers to the service.
Business Goal 02) Guarantee a fair management of taxi queues.
Passengers can request a taxi either through a web application or through a mobile app. The system answers to the request by informing the passenger about the code of the incoming taxi and the waiting time.
Taxi drivers use a mobile application to inform the system about their availability and to confirm that they are going to take care of a certain call.
myTaxiService will provide general functionalities for managing:

- **Profile**
  myTaxiService will manage personal data of the different types of users. Users must be registered to use the service, both for daily users and administrators.
- **Connections**
  myTaxiService will manage the network of connections between users and taxi drivers.
- **Taxis distribution**
  myTaxiService system guarantees a fair management of taxi queues and distribution in the city.
- **Users**
  myTaxiService will manage registering, logging in/out users.

2.3 REFERENCES

1. Analysis document: RASDv5.pdf
2. Document Design: DDv3.pdf
3. Alloy document: 2015_project_Alloy_myTaxiService.als
4. Integration plan document: Integration Plan Document.pdf

2.4 REVISION HISTORY

This paragraph records all revisions to the document.

2.4.1 RASD revisions

- Version 1: initial paper.
- Version 2:
  1. Adjustments of the class diagram. The field “taxi_driver_code” in taxi driver class has been canceled. The field “taxi_arrival_time” has been moved from taxi class to request class.
  2. Adjustments of the functional requirements. [FR10] has been modified in order to support a better management of system user and login
- Version 3:

1. Section 4.1.5 “Planning chart” was added. It contains a Gantt chart, which describes the project’s phases.

- Version 4:
  1. Introduction of [FR31]
- Version 5:
  1. Removal of Gantt Chart

2.4.2 Document Design revisions

- Version 1: initial paper.
- Version 2:
  1. Gantt chart moved in Rasd document version 3.
  2. Section “Other design decision” modified. Now it describes the management of GPS tracking and taxi position in the system.
  3. In the “Runtime view” a sequence diagram, which describes the creation of a new request, has been added.
- Version 3:
  1. Modification of the interfaces between the components in the application server [see chapter 3.2.3.2]
  2. Introduction of Google Maps API component, in the project architecture.

3 COST ESTIMATION

In this section estimates are made to analyse cost, effort and resources involved in the project. Approximate analysis can only be performed, because a lot of components interact in costing analysis:

- Hardware and software
- Travel and training costs
- Effort costs (salaries, social and insurance costs…)
- Overheads (costs of building, heating, lighting, networking, communications…)
- Social, economic, political and business consideration influence the price charged.

The team approaches to this estimation using an algorithmic cost modelling:

1) Function-points (FP) estimation
2) COCOMO (Constructive Cost Model) estimation

3.1 ALGORITHMIC ANALYSIS: FUNCTION-POINTS (FP)

Function points were defined in 1979 by Allan Albrecht at IBM. The functional user requirements of the software are identified and each one is categorized into one of five types: outputs, inquiries, inputs, internal files, and external interfaces. Once the function is identified and categorized into a type, it is then evaluated for complexity and assigned a number of function points.
This is the formulation of the Unadjusted Function Points (UFP)

$$UFC = \sum(number\ of\ elements\ of\ given\ type) * (weight)$$

It is much better to use UFP in combination with other approaches (e.g. COCOMO), instead of computing the adjusted FP. So we proceed using this correction.

3.1.1 Weighting function points

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

3.1.2 Function types

The functional requirements of the software are categorized into one of five types: outputs, inquiries, inputs, internal files, and external interfaces.

3.1.2.1 Internal Logical File (ILF)

Homogeneous set of data used and managed by the application.

The application stores the information about:

1. User.
   Entity dedicated to store information about the user.
2. Taxi.
   Entity dedicated to store information about the taxis.
3. Request.
   Entity dedicated to store information about the requests.
4. Reservation.
   Entity dedicated to store information about the reservations.
5. City Area.
   Entity dedicated to store information about the city area.
6. Position.
   Entity dedicated to store information about the position in the city.

Each of these entities has a simple structure as it is composed of a small number of fields. Thus, the team decide to adopt for all the six the simple weight.

Thus, 6 * 7 = 42 FPs concerning ILFs.

3.1.2.2 External Interface File (EIF)

Homogenous set of data used by the application but generated and maintained by other applications.

The application manages the interaction with the Google Maps API system for acquiring information about a certain position:

1. Google Maps API output in JSON.

Just one entities with a simple structure. Thus we decide to adopt a simple weight for it. We get 1* 5 = 5 FPs.

3.1.2.3 External Input

Elementary operation to elaborate data coming from the external environment.

- The application interacts with all the user as follows:
  - Login/logout: these are simple operations, so we can adopt the simple weight for them. 2 x 3 = 6 FPs

- The application interacts with the passenger as follows:
  - Modify their profile. This operation involves few entities in the system. We can adopt the simple weight. 1 x 3 = 3 FPs
  - Remove the account. This operation involves few entities in the system. We can adopt the simple weight. 1 x 3 = 3 FPs.
  - Create, delete, and modify a reservation. These are complex operation because they involve more entities. As such, we can considered them of high complexity. 3 x 6 = 18 FPs
  - Create, delete, and modify a request. These are complex operation because they involve more entities. As such, we can consider them of high complexity. 3 x 6 = 18 FPs
- The application interacts with the taxi driver as follows:
  - Modify their profile. This operation involves few entities in the system. We can adopt the simple weight. 1 x 3 = 3 FPs
  - Remove the account. This operation involves few entities in the system. We can adopt the simple weight. 1 x 3 = 3 FPs.
  - Accept or decline a taxi request/reservation. These operations involves more entities. As such, we can consider them of high complexity. 2 x 6 = 12 FPs.

- The application interacts with the administrator as follows:
  - Change queue configuration. This operation involves more entities. As such, we can consider them of high complexity. 1 x 6 = 6 FPs.
  - Block an account. This is a simple operation, so we can adopt the simple weight for them. 1 x 3 = 3 FPs
  - Create or delete a new taxi zone. This operation involves more entities. As such, we can consider them of high complexity. 2 x 6 = 12 FPs.
  - Add a taxi driver to the system. This operation involves a limited number of entities. We can adopt the medium weight for it. 1 x 4 = 4 FPs.

In summary, we have a total of 91 FPs.

3.1.2.4 External Output

Elementary operation that generates data for the external environment.

- The application allows system to generate:
  - Notifications to the taxi driver (for instance when he/she has been assigned to a certain drive).
  - Notifications messages for the customers (for instance when the taxi driver accept the drive).

These two external output operations involve just few entities in the system and are not really difficult to implement. As such we consider them of simple complexity. 2 * 4 = 8 FPs

3.1.2.5 External Inquiry

Elementary operation that involves input and output.

- The application allows passenger to request information about:
  - Their profiles.
  - The list of requests and reservations.
- The application also allows the administrator to visualize the information about:
  - All the customer.
  - Taxis.
  - City areas and area's queues.

    - All the request and reservations.
- The application also allows the taxi drivers to visualize the information about:
    - Their profiles
    - Their incoming requests and reservations.

In summary, we have 8 different external inquiries that we can consider of simple complexity, because they interact with an average number of entities in the project and are not so complex to implement. Thus, 8 x 3 = 24 FPs.

At the end the sum of all the partial points gives an

$$UFP = 42 + 5 + 91 + 8 + 24 = 170 \text{ FPs}$$

FPs can be used to estimate LOC depending on the average number of LOC per FP for a given language. We assume that the main language used in developing is J2EE. The equation is the following:

$$LOC = AVC * \text{number of function points}$$

In the case of Java language, AVC is in average 46. The coefficient was taken from: http://www.qsm.com/resources/function-point-languages-table (we used the Avg column)

In this casa the estimate number of lines of code are:

$$LOC = 46 * 170 = 7820 \text{ LOC}$$

3.2 Algorithmic Analysis: COCOMO (COnstructive COst MOdel)

The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry W. Boehm. The model uses a basic regression formula with parameters that are derived from historical project data and current as well as future project characteristics.

The COCOMO II calculations are based on estimates of a project's size in Source Lines of Code (SLOC).

Basic hypotheses of COCOMO:

- The development process is assumed to follow the traditional waterfall scheme.
- Requirements are considered mainly stable for all the project duration.
- The architectural design is accomplished by a small high quality team deeply knowledgeable about the application problem.
- Detailed design, development and unit test are performed in parallel on different sub-systems by different teams of programmers.
- Documentation is written incrementally during the project lifetime.
- Modern project management techniques are used throughout the project.

These preconditions are not completely satisfied by our team, nevertheless we try to perform an estimate to evaluate the project’s effort and duration.

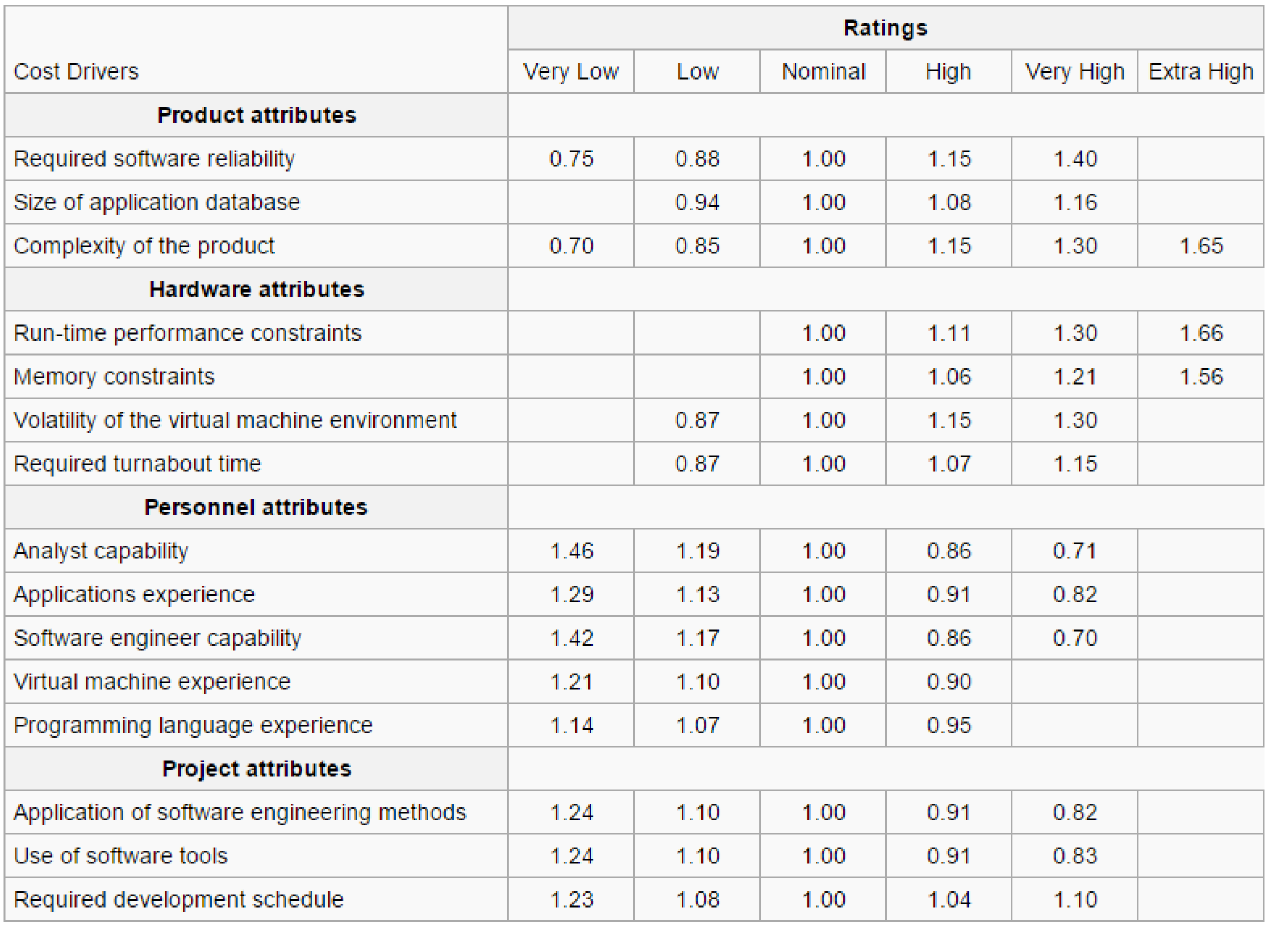
3.2.1 Effort Equation

COCOMO II model makes its estimates of required effort (measured in Person-Months – PM) based primarily on personal estimate of the software project's size (as measured in thousands of SLOC – KSLOC, i.e. lines of codes):

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

EAF: Effort Adjustment Factor derived from Cost Drivers (product of the effort multipliers corresponding to each of the cost drivers for your project). E: Exponent derived from Scale Drivers.

3.2.1.1 EAF (Effort Adjustment Factor) Computation



Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

This is the list of estimated cost driver, according to the personal skills and the experience of the team. All the value are between the range of LOW – NOMINAL due to the fact that the group has limited experience in software engineering project and in developing in J2EE environment.

Cost Drivers	**Ratings**	
	Point	**Complexity**
Product attributes		
Required software reliability	1	Nominal
Size of application database	0.94	Low
Complexity of the product	1	Nominal
Hardware attributes		
Run-time performance constraints	1	Nominal
Memory constraints	1	Nominal
Volatility of the virtual machine environment	0.87	Low
Required turnabout time	0.87	Low
Personnel Attributes		

Analyst capability	1	Nominal
Applications experience	1	Nominal
Software engineer capability	1.17	Low
Virtual machine experience	1.21	Low
Programming language experience	1	Nominal
Project attributes		
Application of software engineering methods	1.1	Low
Use of software tools	1.1	Low
Required development schedule	1.08	Low

The product of all effort multipliers results in an effort adjustment factor (EAF).

$$EAF = \prod(point) = 1.316275$$

This high value of the EAF derives from the limited capacity of the team in developing this kind of project with such technology.

3.2.1.2 Exponent E

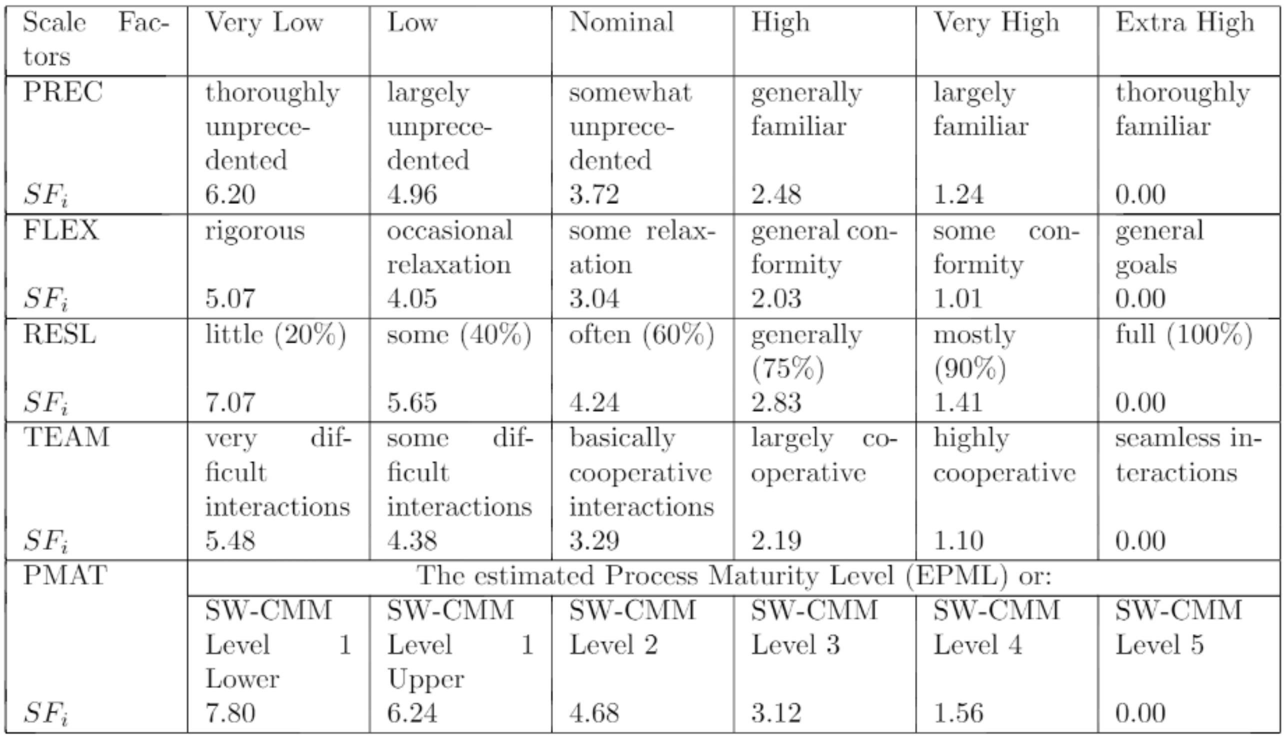
The exponent E is an aggregation of five scale factors (SF) that account for the relative economics or diseconomies of scale encountered for the software projects of different sizes.

The exponent E is calculated as follows:

$$E = B + 0.01 * \sum_{j=1}^{5} SF\,j$$

Where B = 0.91 (for COCOMO II).

The team used this following table to retrieve the weights of each SF.



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_i	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_i	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_i	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_i	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	The estimated 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_i	7.80	6.24	4.68	3.12	1.56	0.00

This is the list of appropriate scale factors rating levels for our project:

Scale Factor	Rating level	Description
Precedentedness (PRECC)	Nominal	The team never developed a web-based project and did not have lot of experience with J2EE environment, but has some previous experience on Java developing.
Development Flexibility (FLEX)	High	The only constraints we had were on the language to be used on functional requirement that the system has to fulfil.
Architecture / Risk Resolution (RESL)	Nominal	We devoted quite a lot time to establish the system architecture given the general product objective. Nevertheless it was difficult to predict risks at this state of the art.
Team Cohesion (TEAM)	Very High	Stakeholder had shared objectives and were committed to reach them. The team had already worked on other projects and so it was rather easy to organize the work.
Process Maturity (PMAT)	Nominal	We chose level 2 because processes have been planned and documented at the project level.

Having chosen the rating levels for each scale factor we can now calculate the exponent:

$$E = 0.91 + 0.01 * (3.72 + 2.03 + 4.24 + 1.1 + 4.68) = 1.0677$$

This value is mainly due to the fact that, despite the lack of previous experience, the team was small and easy to organize.

3.2.2 Estimation Analysis

The following results can be obtained by varying the dimension of KLOC estimated.

KLOC	Effort (person-months)	Duration (months)	Number of person
5.083	21.96	10.97	2
7.820	34.78	17.39	2
10.557	47.92	23.96	2

The Duration column has been obtained in this way: Effort/Number of person = Duration.

Starting from these results, the manager can decide, accordingly to the cost constraints and to the availability of people to be employed in the future, the size of the application (lower value of KLOC for example can be selected deciding to give low priority to some aspects of the application and deferring them to a future release) that can better satisfy the overall requirements of the application.

We have also provided two different evaluation considering a variation of +/- 35 % of the KLOC, to have an optimistic and pessimistic vision of the effort and the time.

4 PROJECT SCHEDULING

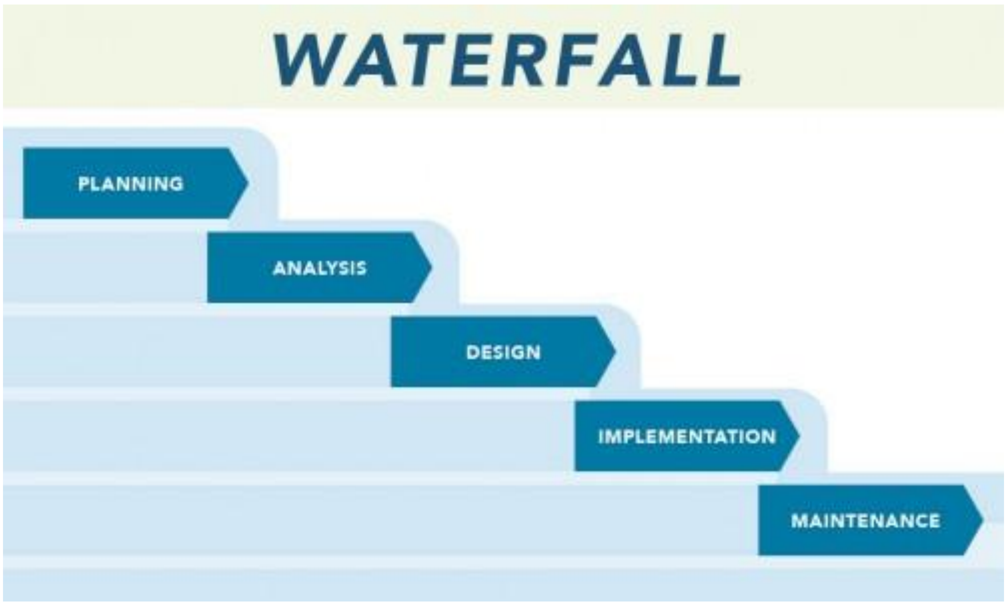
This section identify the tasks of the project and their schedule.

4.1 THE WATERFALL METHODOLOGY

Waterfall is a linear approach to software development. In this methodology, the sequence of events is something like:

1. Gather and document requirements
2. Design
3. Code and unit test
4. Perform system testing
5. Perform user acceptance testing
6. Fix any issues
7. Deliver the finished product

In a Waterfall development project, each of these represents a distinct stage of software development, and each stage generally finishes before the next one can begin.



4.1.1 Advantages

This is a list of pros that the team has analysed in order to select this kind of strategy.

- Developers and customers agree on what will be delivered early in the development lifecycle. This makes planning and designing more straightforward.
- Progress is more easily measured, as the full scope of the work is known in advance.
- Throughout the development effort, it's possible for various members of the team to be involved or to continue with other work, depending on the active phase of the project. For example, business analysts can learn about and document what needs to be done, while the developers are working on other projects. Testers can prepare test scripts from requirements documentation while coding is underway.
- Except for reviews, approvals, status meetings, etc., a customer presence is not strictly required after the requirements phase.
- Because design is completed early in the development lifecycle, this approach lends itself to projects where multiple software components must be designed (sometimes in parallel).

- Finally, the software can be designed completely and more carefully, based upon a more complete understanding of all software deliverables. This provides a better software design.

Some disadvantages may arise using this kind of methodology, which is not always very flexible. For these reason a more experienced and skilled team, maybe with a higher number of members, would have preferred a different strategy. For example an Agile methodology could be more efficient in this kind of project. By the way, we have preferred to use this strategy because more common for the team and perhaps easier for beginners developers.

4.2 Schedule and Milestones

The following table depicts the way milestones are coupled to various project phases and when they are scheduled:

- The team budget of 2 persons x 1470 hours = 2940 hours;
- The project deadline of July 23th 2017
- The final presentation of July 14th 2017
- The intermediate presentation of January 11th 2016
- The peer evaluation deadline of November 26th 2015
- A holiday from December 26th till January 6th
- A holiday from February 27th till March 3th

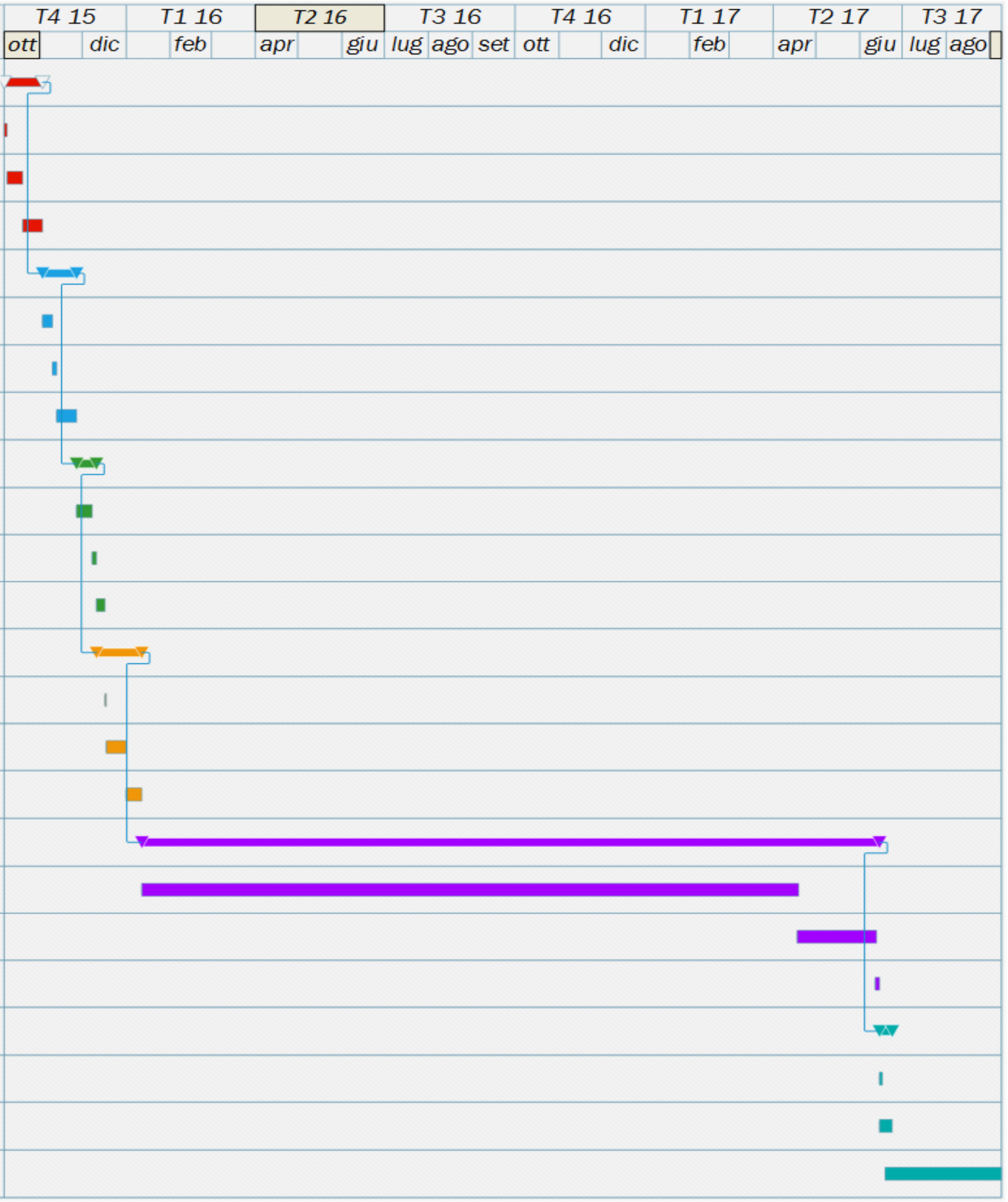
This table provides also indicative dates in which deliverables are validated and distributed.

Phase	Milestone	Description	Deliverables	Planned Date
Start project	M1	Project kick-off		7/10/2015
User Requirements		Analysis Document Approved	Planning document	19/10/2015
	M2	User requirements document approved	RASD (Requirements document)	02/11/2015
Software Requirements		Prototype Approved		26/11/2015
	M3	Software requirements document approved		16/12/2015
Architectural Design Document	M4	Architectural Design Document approved	Document Design – Test Plan document	11/10/2016
Detailed Design	M5	Coding complete	Codes and tests	18/04/2017
Transfer phase	M6	Acceptance test successful		14/06/2017
	M7	Product delivery complete	Finished product.	23/6/2017

4.3 Gantt Chart

This section provides a Gantt chart which describes the tasks of the project and their schedule.

ID	Activity	Start	End	Duration
1	**Planning**	**07/10/2015**	**02/11/2015**	**19g**
2	Problems analysis	07/10/2015	08/10/2015	2g
3	Domain analysis	09/10/2015	19/10/2015	7g
4	RASD and project planning	20/10/2015	02/11/2015	10g
5	**Requirements Analysis**	**03/11/2015**	**26/11/2015**	**18g**
6	Functional and non-functional requirements documentation	03/11/2015	09/11/2015	5g
7	Use cases analysis	10/11/2015	12/11/2015	3g
8	Activity diagram, sequence diagram, UX	13/11/2015	26/11/2015	10g
9	**Detailed analysis**	**27/11/2015**	**10/12/2015**	**10g**
10	Definition of functionalities and system limits	27/11/2015	07/12/2015	7g
11	Software modules analysis	08/12/2015	10/12/2015	3g
12	Logic layer analysis	11/12/2015	16/12/2015	4g
13	**Design**	**11/12/2015**	**11/01/2016**	**22g**
14	Project activities analysis	17/12/2015	17/12/2015	1g
15	Presentation design and service design	18/12/2015	31/12/2015	10g
16	Physical components analysis	01/01/2016	11/01/2016	7g
17	**Development**	**12/01/2016**	**14/06/2017**	**372g**
18	Coding	12/01/2016	18/04/2017	331g
19	Integration & system test	18/04/2017	12/06/2017	40g
20	Software installation	12/06/2017	14/06/2017	3g
21	**Deployment**	**15/06/2017**	**23/06/2017**	**7g**
22	System activation	15/06/2017	16/06/2017	2g
23	User training	15/06/2017	23/06/2017	7g
24	Maintenance	19/06/2017	08/09/2017	60g



4.4 Task, durations and dependencies

4.4.1 Work Packages

This table summarize the main tasks executed during the project. An estimate effort and duration has been developed, analysing the COCOMO result, the number of members in the team and the expected duration of the task. Trivially the effort has been calculate using this formula Effort = Duration * (Number of people working on the task). For the planning and the analysis, we estimate an effort which is less with respect to the coding. This is due to the fact that part of the analysis have already been done. Developing part will be trickier because of the inexperience of the team, the risks which we have to face during this tasks and the complexity of the requirements to develop.

Task	Effort (person-days)	Duration (days)
Planning	38	19
Requirement Analysis	36	18
Detailed Analysis	20	10
Development	744	372
Deployment	14	7
Maintenance	120	60

4.4.2 Dependencies

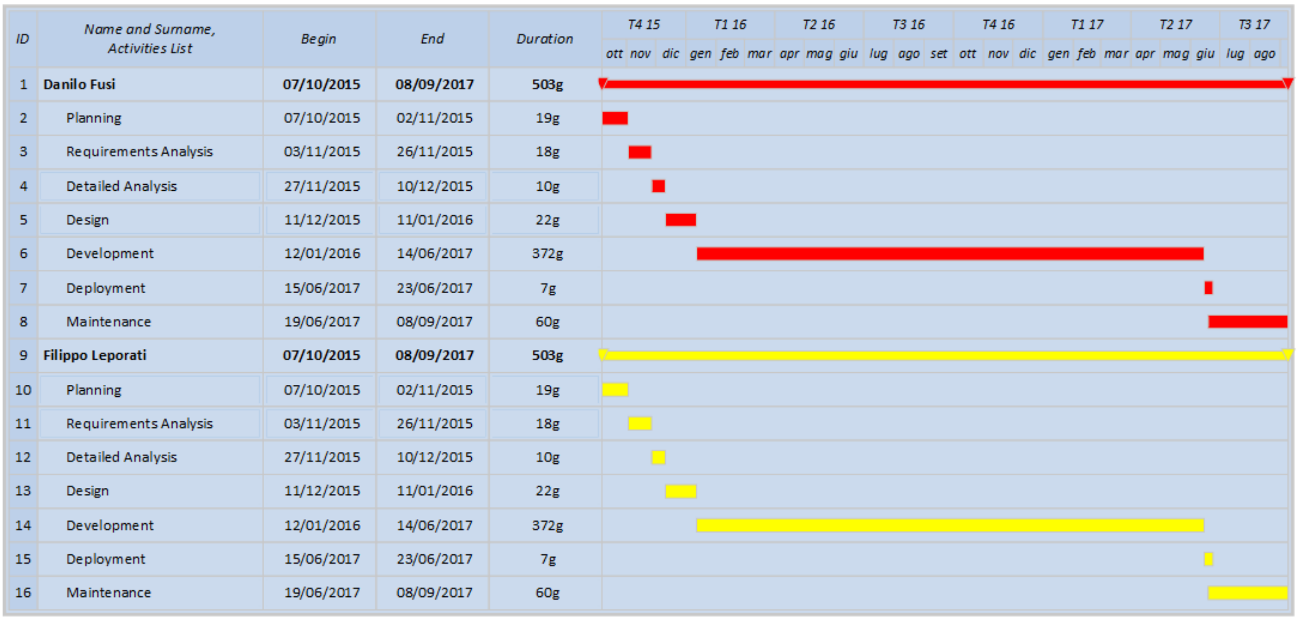
For the User Requirements and the Software Requirements phase, a dependency chart is not necessary as dependencies are trivial. The User Requirements Document must be more or less ready before work on the Software Requirements Document can start. However, it is inevitable that, while working on the Software Requirements Document, new questions arise about the user requirements. So that documents does not strictly depend of the User Requirements Document. Similarly working on the Architectural Design Document can lead to changes to the Software Requirements Document and so on. The prototype must not be later than the Software Requirements Document since it is part of it. Within the Architectural Design phase no work packages are dependent on each other. When the first results of the Architectural Design phase are there, the Detailed Design phase commences. Work packages don't depend on each other since interfaces between components have been defined in previous phases. The Deployment phase is the last phase. Obviously, it is not possible to transfer the product before it is ready. Therefore the Detailed Design phase must be completed before the Deployment phase starts.

5 TEAM ORGANIZATION

5.1 STAFF ALLOCATION CHART

The following chart explains how the different tasks, described in the Gantt chart, are divided between our project team, composed by only two people.

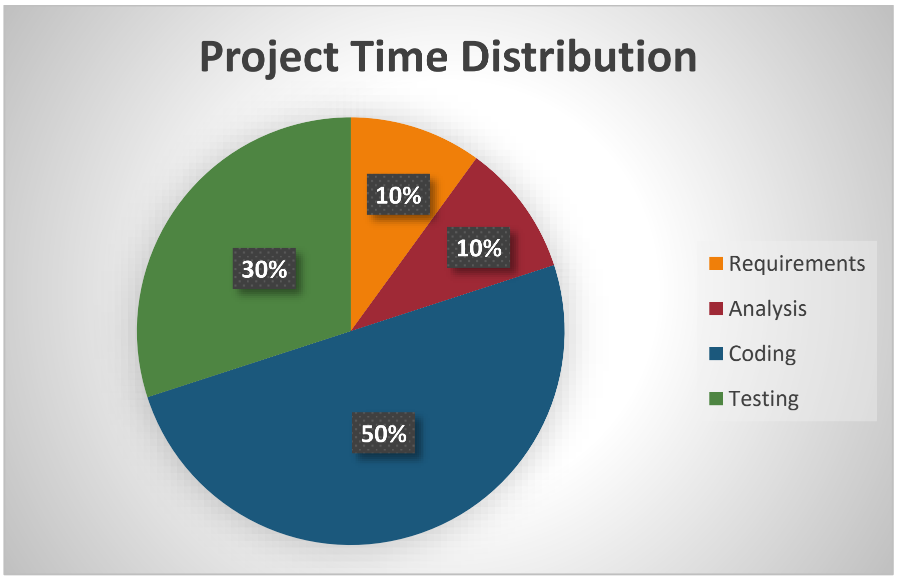
In this small group of two people, it is not necessary the presence of a Project Manager. This is due to the fact that the group is very limited and communication problem does not exist. Project decisions can be taken efficiently and in short time, simply meeting and discussing in group.



ID	Name and Surname, Activities List	Begin	End	Duration
1	**Danilo Fusi**	**07/10/2015**	**08/09/2017**	**503g**
2	Planning	07/10/2015	02/11/2015	19g
3	Requirements Analysis	03/11/2015	26/11/2015	18g
4	Detailed Analysis	27/11/2015	10/12/2015	10g
5	Design	11/12/2015	11/01/2016	22g
6	Development	12/01/2016	14/06/2017	372g
7	Deployment	15/06/2017	23/06/2017	7g
8	Maintenance	19/06/2017	08/09/2017	60g
9	**Filippo Leporati**	**07/10/2015**	**08/09/2017**	**503g**
10	Planning	07/10/2015	02/11/2015	19g
11	Requirements Analysis	03/11/2015	26/11/2015	18g
12	Detailed Analysis	27/11/2015	10/12/2015	10g
13	Design	11/12/2015	11/01/2016	22g
14	Development	12/01/2016	14/06/2017	372g
15	Deployment	15/06/2017	23/06/2017	7g
16	Maintenance	19/06/2017	08/09/2017	60g

T4 15
T1 16
T2 16
T3 16
T4 16
T1 17
T2 17
T3 17
ott nov dic gen feb mar apr mag giu lug ago set ott nov dic gen feb mar apr mag giu lug ago

Since the team is composed by only two people, we have not decided to split the tasks and all of them are accomplished simultaneously. The team will work always together in order to improve project's efficiency and to have a continuous communication between the members. This kind of teamwork should reduce the misunderstandings in the group and the errors that may occur during the project. Moreover, the efficiency should increment because, in case a risk arises, the team can always face it together.
The pie chart below shows also the project time distribution:



5.2 Meeting and Communication

This section describes the principles for reporting and distributing information within the project for the internal and external stakeholders. Internal and external communication are described in the following table.

Type of communication	Method / Tool	Frequency/Schedule	Information
Internal Communication			
Project Meetings	Conference	Weekly and on event	Project status, problems, risks, changed requirements
Sharing of project data	Shared Project Server	When available	All project documentation and reports
Milestone Meetings	Conference	Before milestones	Project status (progress)
Final Project Meeting	Conference	After milestone M7	Wrap-up Experiences
External Communication			
Project Report	Excel sheet	Monthly	Project status - progress - forecast - risks
Stakeholder Meetings	Conference	Monthly	Report project information

6 Risk Management

This section describes the possible risks of the project. Also, the possible actions for correcting the risks are explained.

We have identified the following categories of risks:

1) Project Risks
2) Technical Risks
3) Business Risks

For each risk, a description, a probability to occur, the action associated and the impact of the risk are given.

6.1 Project Risks

We have reported only the important project risks.

6.1.1 Problem of communication

Probability: Medium

Prevention: After a meeting, one person creates a report. Every participant and every person who should have been a participant of the meeting should get a copy of this report. Team members should not hesitate to ask and re-ask questions if things are unclear.

Correction: When there is a problem of communication, a new meeting is arranged in order to clarify.

Impact: High

6.1.2 Lack of time
Probability: High

Prevention: Care is taken when the time schedule is planned.

Correction: When tasks fail to be finished in time or when they are finished earlier than planned the project planning is adjusted. If lack of time becomes severe, the application is released as a beta version, after a consultation with the costumer.

Impact: High

6.1.3 Illness or absence of team members
Probability: High

Prevention: Team members should timely before a planned period of absence.

Correction: Since the team is composed by only two members and the illness period can't be forecast, when the time schedule is planned, it is needed to extend the delivery forecast.

Impact: High

6.1.4 Lack of knowledge of team members
Probability: Low

Prevention: It must be considered the technical knowledge needed for developing the project.

Correction: If there is a lack of knowledge of a team member, it is necessary to train him/her for learning the knowledge necessary.

Impact: Medium

6.1.5 The customer changes his mind about the requirements
Probability: High

Prevention: It is obviously explained to the customer, that after he has accepted a version of the RASD, the RASD cannot be changed by the customer's wish only.

Correction: If the customer changes his mind during the development phase, his new requirements can be incorporated in the RASD.

Impact: Medium

6.1.6 The customer is not available when needed
Probability: Medium

Prevention: Meetings with the customer can be planned well in advance.

Correction: When the customer is not available, meetings may have to be rescheduled.

Impact: Medium

6.2 Technical Risks
We have reported only the important technical risks.

6.2.1 Design Errors
Probability: Medium

Prevention: The design of the application should be made carefully.

Correction: When errors in the design of the application are noticed, should be correct as soon as possible. Also all the work, that depends on the faulty design, should be halted until the error is corrected.

Impact: High

6.2.2 Server crash

Probability: Low

Prevention: All products are stored in the project repository, which is backed up regularly.

Correction: When a product is lost, it is recovered from the most recent backup.

Impact: Medium

6.2.3 User Interface mismatch

Probability: Low

Prevention: All User Interface are designed in advance in order to avoid any kind of mismatching.

Correction: When there is a user interface that is not suitable for the end-user, it is designed again.

Impact: Low

6.2.4 Component is complex to implement

Probability: Low

Prevention: A deep analysis of the product to develop must be done before the beginning of the development phase.

Correction: If a component is too complex to implement, maybe it can be split in other subcomponents in order to make the components easier to implement.

Impact: Medium

6.3 Business Risks

We have reported only the important business risks.

6.3.1 Wrong budget estimation

Probability: Low

Prevention: Care is taken when the budget for this project is estimated.

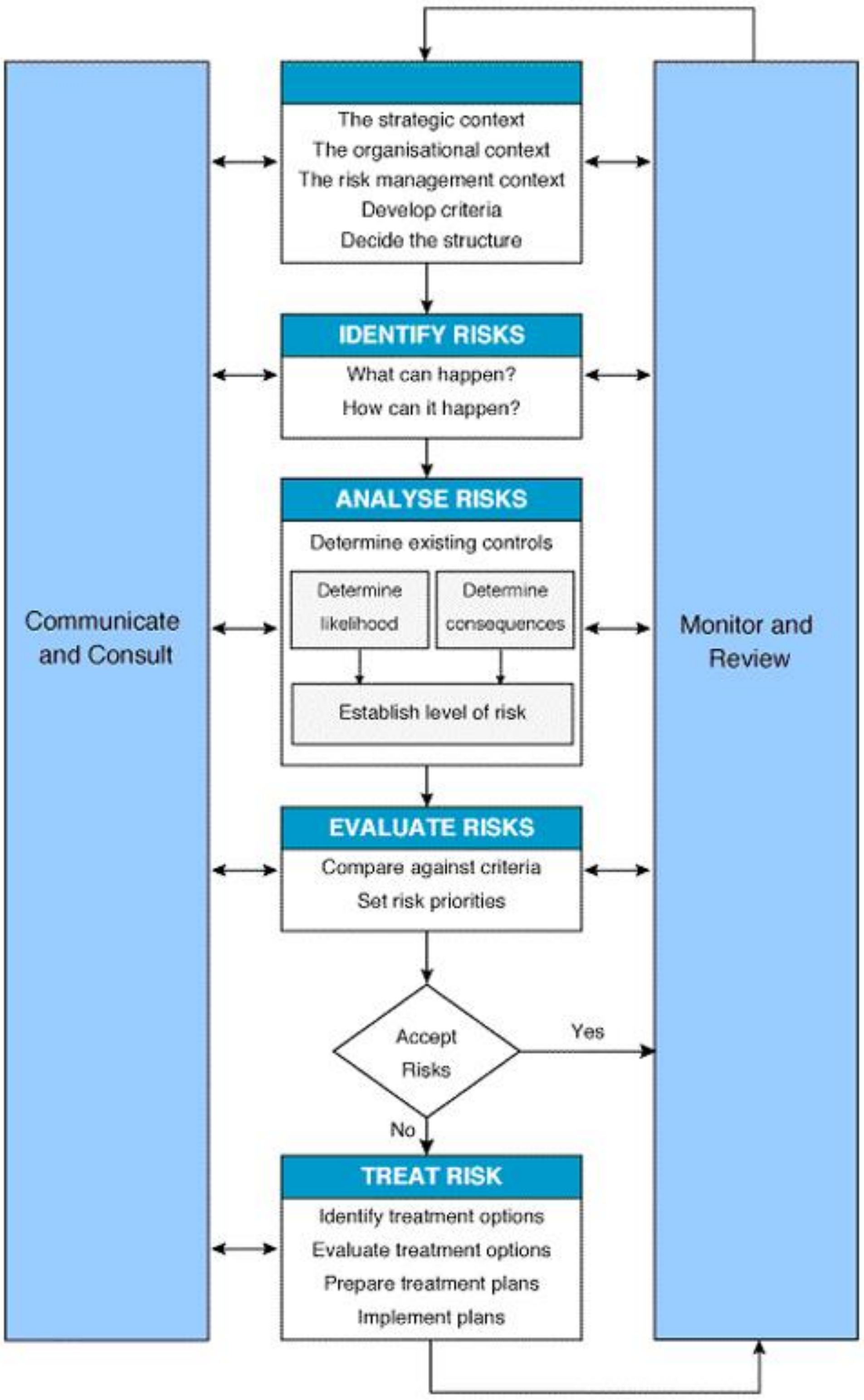
Correction: When the real budget needed is greater than the budget previously estimated, a new meeting with the costumer is planned in order to make the decision of make available a new amount of budget or not.

Impact: Medium

6.4 The risk Management Process

The risk management process is a set of procedures and practices for establishing, identifying, treating and monitoring all the risks about the project. The risk management contribute to identify the impact that could have the risks on the project.

The diagram below show how the risk management process works, explaining all the main phases that characterized it.



The risk management process is characterized by the following phases:

1. **Identify the risks:** this is the initial phase that can identify all the risks that can occur during the developing of the project.
2. **Identify the causes:** this phase aim to identify all the causes that can cause the risk.
3. **Analysis phase** that includes:
    a. **Likelihood estimation:** estimate the probability that a risk will occur. Generally, it is not a precise number but is enough generic (e.g. Low, Medium, High).
    b. **Consequences identification:** describe all the possible consequences that can happen when a risk occur, and how can influence the development of the project.
    c. **Risk level identification:** rate the level of the risk (e.g. Low, Medium or High).
4. **Risk Treatment:** the treatment of the risk includes all the possible solution that can be applied for resolving it.
5. **Monitor and Review:** monitoring of all risks and regular review of them.

6.5 Summary

It is obvious that problems will occur during the project. To avoid problems the following rules should be followed by all team members:

- Try to signal problems as early as possible and report them to the other colleague, so that action can be taken;
- Pay attention to communication and make sure everybody understands the things the same way;
- Focus on the agreed user requirements, which express the wishes of the customer;
- Minimize friction between team component by helping and supporting each other;
- Follow guidelines that are posed in this document to aid coordination and to ensure product quality