Salesmen (Group2 SE)

Software Project Management Plan

author(s): Nick De Cooman

Abstract: This document details the primary management details of the Salesmen project. This project is created by group 2 of the course Software Engineering in the 2009-2010 study year at the Vrije Universiteit Brussel.

21-12-2009

History of versions

Version	Date	Author(s)	Changes
v0.1.0	21-10-2009	N. De Cooman	Initial draft
v0.1.1	22-10-2009	N. De Cooman	Revision after quality control
v0.2.0	12-11-2009	N. De Cooman	Revision including customer's comments
v0.2.1	20-11-2009	N. De Cooman	Adding estimation of effort
v0.2.2	22-11-2009	N. De Cooman	Revision on process model, adding real costs
v0.2.3	23-11-2009	N. De Cooman	Adding Gantt chart
v0.2.4	26-11-2009	N. De Cooman	Adding estimation of costs
v0.3.0	10-12-2009	N. De Cooman	Updating monitoring mechanisms
v0.3.1	12-12-2009	N. De Cooman	Updating costs, updating Gantt chart
v0.4.0	21-12-2009	P. Provinciael	Made the document consistent with the other documents

Contents

Chapter 1

Introduction

1.1 Project overview

The goal of this project is to design and implement the software behind an online auction website such as eBay. The codename of our project is *S*alesmen.

This project is embedded in the Software Engineering course, which is designed for senior students of Computer Science and Engineering at the Vrije Universiteit Brussel. It has a threefold purpose:

1. Get familiar with the software development process and learn to work in a team.
2. Develop a system that supports the functionality of an auction website. This means that a user will be able to sell or buy items in an online auction.
3. Develop and maintain a general website on our project. Up to date versions of all documents such as minutes, timesheets and deliverables will be available at any time. This site will be hosted at `http://wilma.vub.ac.be/~se2_0910/`.

The software of the application will be written in Java and distributed under an Open Source license.

1.2 Project deliverables

- Software documentation (see 4.2)
- Minutes: 3 days after meeting
- Timesheets: every Monday before 12H00
- Source code

All of those documents will be available in English on our website.

1.3 Reference materials

See Appendix A.

1.4 Definitions and acronyms

SCMP	Software Configuration Management Plan
SDD	Software Design Plan
SPMP	Software Project Management Plan
SQAP	Software Quality Assurance Plan
SRS	Software Requirements Specification
STD	Software Test Document
COCOMO	Constructive Cost Model

Chapter 2

Project Organization

2.1 Process model

The chosen proces model will be the spiral model, with 2 iterations. Unlike the waterfall process, we will be able to show partial versions to the customer for feedback, reducing risks such as faulty requirements. The application will consists of some basic functionality after the first iteration and more advanced features will be implemented during the second one.

A detailed planning of this process can be found in section 5.3.

2.2 Organizational structure

The structure of our team will be organised as followed:

2.2.1 Project Manager

- Head of the project team
- Deliver the SPMP
- Risk Analysis
- Keep track of the project's progress
- Manage timesheets
- Prepare and chair meetings
- Spokesman to organizational boundaries and interfaces
- Motivate and improve internal collaboration

2.2.2 Project Secretary

- Resume meetings
- Deliver minutes

2.2.3 Configuration Manager

- Deliver the SCMP
- Manage revision control system
- Install and maintain used tools
- Make backups of all project documents

2.2.4 Quality Assurance Manager

- Head of quality team
- Deliver SQAP and STD
- Design and implement test framework
- Coordinate testing as defined in STD
- Verify the general quality of the product
- Verify the quality of documents

2.2.5 Requirements Manager

- Deliver SRS
- Check whether the requirements are implemented
- Look for extra functional requirements
- Interview customer related to the requirements
- Present final requirements to the customer

2.2.6 Design Manager

- Head of design team
- Deliver SDD based on the SRS
- Manage design of the system
- Check whether implementation is based on the SDD
- Design and manage database according to SRS and SQAP
- Present design to the customer

2.2.7 Implementation Manager

- Head of implementation management
- Manage the integration of different parts of the code
- Track errors
- Define implementation proces based on the SDD
- Manage documentation of code

2.2.8 Webmaster

- Develop and maintain project website
- Upload documents, after approval of the project manager

2.3 Organizational boundaries and interfaces

The external communication with the customer will only happen via the project manager, the requirements manager and the design manager. The project manager reports major problems, progress status and other organisatorial issues. The requirements manager discusses the functionality of the system and the design manager will present the design of the system to the customer.

2.4 Project responsibilities

The responsibilities, as defined in 2.2, will be associated to the following persons:

Role	Effective	Assistant
Project Manager	Nick	Patrick
Project Secretary	Jonathan	Wouter
Configuration Manager	Jorne	Sina
Quality Assurance Manager	Patrick	Bart
Requirements Manager	Wouter	Jonathan
Design Manager	Bart	Nick
Implementation Manager	Sina	Jorne
Webmaster	Sina	Wouter

However, this does not mean that a person is only responsible for his/her own function. Every team member can be asked for assistance in all processes in function to achieve the end goal.

A person responsible for creating a deliverable also needs to be aware of the fact that

- Documents will be writen in LaTeX-format, using the provided template.
- Documents need to be ready in time (as defined in 4.2), and sent to the project manager for approval.
- Documents needs to be up-to-date during the entire project.

Chapter 3

Managerial process

3.1 Objectives and priorities

During the development of the system, the following objectives should be kept in mind:

1. The project should meet the requirements of the customer
2. Deadlines have to be respected
3. Reusability and extensibility of the software is very important
4. The system should be stable

3.2 Assumptions, dependencies and constraints

- No assumptions will be made.
- Dependencies
  For the project to succeed, it will depend on the knowledge and the motivation of the team members
- Constraints:
  - Only open-source software is to be used
  - The product should run in a Linux environment
  - The product should run on the Wilma server

3.3 Risk Management

Developing software introduces a lot of risks. At regular times these risks will be discussed in order to reduce or eliminate them. Each member is encouraged to report possible risk to the project leader.

3.3.1 Google goes down for a certain period

Criticality: +
We need to make daily backups of our files and documentation. This will be performed by the configuration team.

Risk	Change	Effect	Disposal	**Priority**
Google down	2	8	8	**216**
Lack of knowledge	6	8	6	**90**
Illness	6	4	3	**105**

Table 3.1: Risk table

3.3.2 Lack of knowledge of certain tools or mechanisms

Criticality: ++++
Some tools and mechanisms will be completely new to some team members which can results in a lack of knowledge and inefficiency. To prevent this, tutorials will be made as Wiki-pages. Moreover, presentations can be given if the subject is too complex or if it demands a complete explanation.

3.3.3 Somebody becomes ill

Criticality: +++
This risk has already been considered in the beginning of this project. This is why there is an Assistant(formerly called Backup) for every Management position. If a leader or manager gets ill, the assistant of that function should be able to fully understand his function and replace that leader or manager, for a certain period of time.

3.3.4 Risk Table

The above risks can be ranked in a table based on the likelihood of the risks, the impact on the project, the cost of disposal and the priority of it. This last one will be calculated as follows :

$$priority = (11 - change) \cdot (11 - effect) \cdot disposal$$

The lower the priority, the most impact it has on the project.

3.4 Monitoring and controlling mechanisms

3.4.1 Meetings

Meetings will be held every week. The topics to handle at the meeting will be defined in an agenda and will be sent to every team member before the start of the meeting. Minutes will summarize the meetings and will be available on the project website within 3 days after the meeting.

Should one not be able to attend the meeting, he/she is requested to inform the project manager within 3 hours before the start of the meeting.

3.4.2 Timesheets

A global timesheet will be available every Monday before 12H00. Team members will need to submit their timesheet on Sunday before 23h59 for approval by the team manager.

3.4.3 SCRUM Report

Every 2 weeks all team members should deliver a SCRUM report in which they have to answer briefly 3 questions:

1. What have you done in the last 2 weeks?
2. What will you do in the next 2 weeks?
3. Is there anything preventing you from doing what you have planned?

In this way, the project manager is able to monitor the global progress of the project and directing certain members where necessary. A summarized version will be available onto the website afterwards.

Chapter 4

Technical Process

4.1 Methods, tools and techniques

- Google Code
  There has been chosen for Google Code because it has some handy features:
  - Revision control system: Subversion in our case, see below.
  - Issue tracker
  - Wiki (is sometimes handy e.g. for meeting agenda).
  - File download server (for our software and document releases).
- Subversion
  This is a centralized version control system chosen for the following reasons:
  - It is easy to refactor the source code structure, while preserving files' history.
  - The whole repository has a single revision that is incremented after each commit.
- Eclipse
  Eclipse is a software development environment comprising an IDE and a plug-in system to extend it. It is used to develop applications in Java. It was decided to work with Eclipse because it has a very userfriendly interface and because of its popularity. Also it's known by most of the team members.
- LaTeX
  LaTeX has been chosen to document this project because it's internationally known and commonly used. Also, a few members were interested in learning this language.

More specific tools will be discussed later.

4.2 Software documentation

Several documents will be publised during the execution of this project.

Document	First version available on
SPMP	October 26, 2009
SCMP	November 9, 2009
SRS	November 13, 2009
SQAP	November 23, 2009
SDD	November 30, 2009
STD	December 07, 2009

4.3 Project support functions

Throughout the entire project, Joeri De Koster and Dirk Vermeir will be available for any help. For Wilma related problems, Dirk Van Deun can be contacted.

Chapter 5

Work packages, schedule and budget

5.1 Costs

5.1.1 Man-costs up to now

Using timesheets enables us to track the real costs of this project. Off course, this only includes man-costs and it does not cover some hidden problems that can occur during the development.

Considering the starter wage of a computer scientist is approximately €2200 gross, we can calculate a realistic earning per hour. Assume one should perform an average of 38 hours per week, and a month consists of 4 weeks. This results in:

$$\text{Wage/hour} = \frac{\text{€2200 wage/month}}{38 \text{ hours/week} \cdot 4 \text{ weeks/month}} \approx \text{€}14,50$$

In that case, the total cost of this project so far is given in table 5.1.

5.1.2 Estimation of total man-costs

Based on the information in table 5.1, we can estimate the total cost of this project by calculating the average number of hours up to now. For the first 8 weeks, the average number of worked hours

Week	Total hours	Total Cost
Week 42	43 hours	€623,50
Week 43	63 hours	€913,50
Week 44	36 hours	€522,00
Week 45	60 hours	€870,00
Week 46	28 hours	€406,00
Week 47	53 hours	€768,50
Week 48	67 hours	€971,50
Week 49	67 hours	€971,50
TOTAL	**417 hours**	**€6046,50**

Table 5.1: Real costs up to now based on timesheets

Parameter	Value
a	3.0
b	1.12
c	2.5
d	0.35

Table 5.2: COCOMO Parameters for a semi-detached project

is *52,1 hours/week*.

Assuming 27 weeks will be needed to accomplish this project, the total number of man-hours can be estimated as:

$$52,1 \text{ hours/week} \cdot 27 \text{ week} \approx 1407 \text{ man-hours}$$

Thus, the total man-cost results in:

$$1407 \text{ man-hours} \cdot 14,50 \text{ euro/hour} = €20.401$$

Note that this is just an estimation, and propably amount to €25.000.

5.1.3 Estimation of effort

To calculate an estimation of costs, the COCOMO algorithm developed by Barry Boehm will be used. Depending on the number of lines of code, the costs and efforts of this project can be computed. Off course, this is just an estimation and will differ from reality. The algorithm provides next formulas in order to calculate the effort and cost:

$$\text{Effort Applied: } E = a \cdot (KLOC)^b$$
$$\text{Development Time: } D = c \cdot E^d$$
$$\text{People required: } P = \frac{E}{D}$$

KLOC is the estimation of the code length expressed in thousands of lines of code. Our project can be classified as a semi-detached project which means that our team of a medium size has a mixed experience working with a mix of rigid and less than rigid requirements. Therefore, the parameters a, b, c, d are defined as in table 5.2.

By analyzing simular projects, we estimate to write 10KLOC. This brings the total effort to:

$$E = 3.0 \cdot 10^{1.12} \approx 39,55 \text{ man-months}$$

The total development time thus becomes:

$$D = 2,5 \cdot 39,55^{0,35} \approx 9,06 \text{ months}$$

This results in a total staff estimation of

$$P = \frac{39,55 \text{ man-months}}{9,06 \text{ months}} \approx 4,37 \text{ people}$$

Using the COCOMO algorithm, we can thus conclude that 4,37 people are needed to realise our project in 9,06 months.

5.2 Dependencies

At this time, it is only possible to provide 1 general dependency :

- Iteration 2 depends on the completion of iteration 1.

5.3 Planning

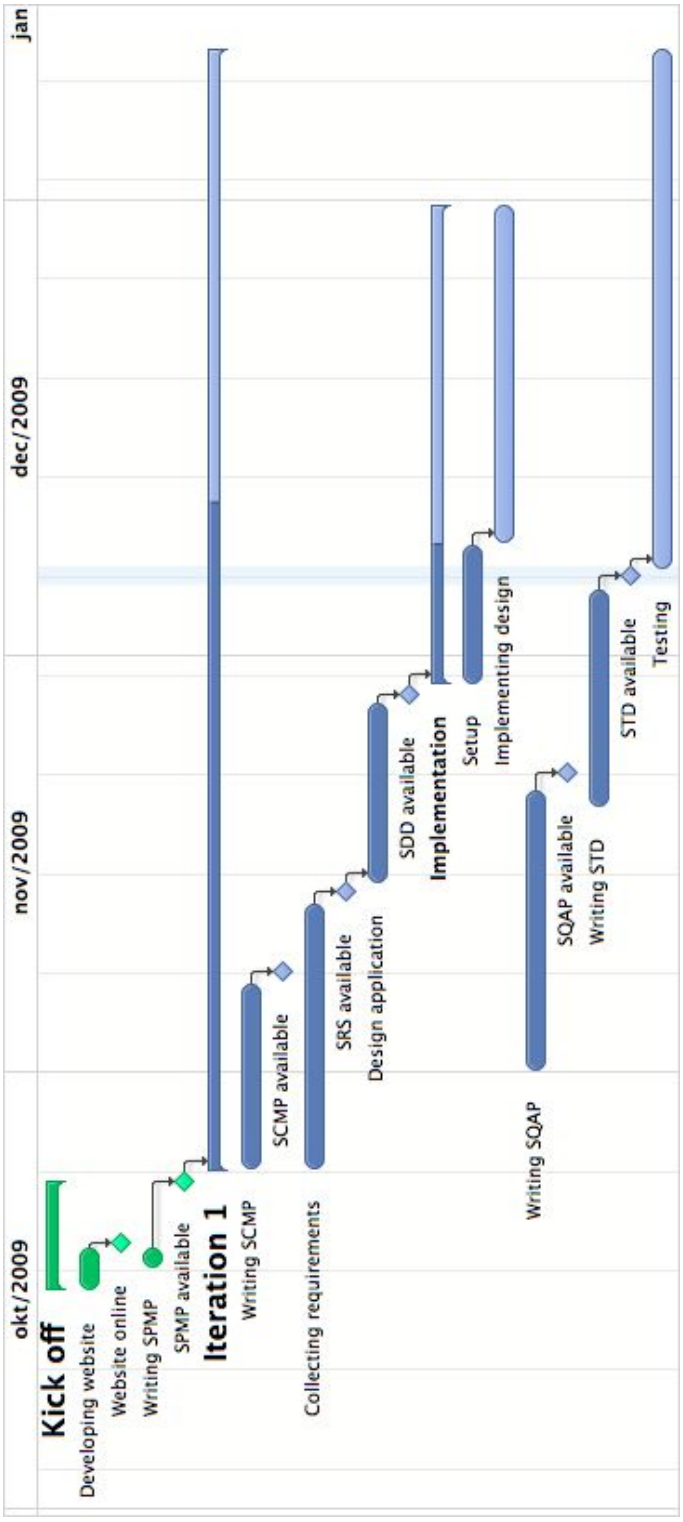

Figure 5.1: Current schedule of planning in form of Gantt-chart

Appendix A

References

[1] Eric J. Braude, *Software Engineering - An Object-Oriented Perspective*, John Wiley & Sons, 2001. ISBN 0-471-32208-3.

[2] Dirk Vermeir *http://tinf2.vub.ac.be/~dvermeir/courses/software_engineering/slides.pdf* 1 Nov 2009.