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.

April 17, 2010

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
v0.3.2	21-12-2009	P. Provinciael	Made the document consistent with the other documents
v0.4.0	15-04-2010	N. De Cooman	Revision after iteration I

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. The 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 project website, which contains all necessary documents and references to deliverables. The 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

The required deliverables are:

- **SPMP** to describe the main management details.
- **SCMP** to describe the configuration and tools.
- **SRS** to describe the software requirements specifications.
- **SDD** to describe the architecture and design.
- **SQAP** to describe the quality assurance guidelines.
- **STD** to describe the testing procedures.
- **Timesheets** to provide an activity log of each week.
- **Minutes** of meetings
- **Source code**

All documents will be available in English on our project 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 is the spiral model, with two iterations. Unlike the waterfall process, we are able to show partial versions to the customer for feedback in order to reduce risks such as faulty requirements. After the first iteration, the application provides some basic functionality and more advanced features are implemented during the second one. A detailed planning can be found in section 5.4.

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
- Check and publish 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

2.2.7 Implementation Manager

- Head of implementation management
- Manage the integration of different parts of the code
- Assign implementation tasks
- Track errors
- Manage documentation of code

2.2.8 Webmaster

- Develop and maintain project website
- Upload documents

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 can present the design of the system to the customer.

2.4 Internal communication

Most internal communication is recommended to happen via a public mailinglist (salesmen@googlegroups.com). Being so, all discussions can be consulted publicly on `http://groups.google.com/group/salesmen/`.

In exceptional situations, private communication is recommended via se2@tinf.vub.ac.be.

2.5 Project responsibilities

2.5.1 General responsibilities

Good agreements make good friends. Therefore, all team members have to be aware of the following responsibilities:

- Timesheets need to be uploaded every week before Sunday 23h59 and should contain sufficient references.
- Documents need to be writen in LaTeX-format, using the provided template and need to be ready in time.
- Meetings should be attended as much as possible.

The subject of monitoring and controlling mechanisms is treated in section 3.4.

2.5.2 Assigned responsibilities

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

Role	Effective	Assistant
Project Manager	Nick De Cooman	Patrick Provinciael
Project Secretary	Jonathan Jeurissen	Wouter Van Rossem
Configuration Manager	Jorne Laton	Sina Khakbaz Heshmati
Quality Assurance Manager	Patrick Provinciael	Bart Maes
Requirements Manager	Wouter Van Rossem	Jonathan Jeurissen
Design Manager	Bart Maes	Nick De Cooman
Implementation Manager	Sina Khakbaz Heshmati	Jorne Laton
Webmaster	Sina Khakbaz Heshmati	Wouter Van Rossem

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

After the first semester, the above division appeared not to be well-balanced based on the timesheets. Therefore, a few changes were made in order to re-balance the timesheets:

- Since Sina Khakbaz Heshmati was occupied with three important aspects, he had much more to do than the other team members. Hence, he worked the double of some others in the first few months, which resulted in disproportionated timesheets.

  Therefore, we deceided that the maintaince of our project website is transferred to Wouter Van Rossem, the assistent webmaster. As regards the configuration, Jorne Laton becomes fully responsible for this.

- As for Jonathan Jeurissen this course only counts for four study-points, there has to be a clear difference in the timesheet balance. Since we had several meetings the first semester, this was obviously not the case. Hence, we deceided that Wouter Van Rossem, the assistent secretary, also becomes responsible for writing the minutes.

Consequently, the current division of tasks is the following:

Role	**Effective**	**Assistant**
Project Manager	Nick De Cooman	Patrick Provinciael
Project Secretary	Wouter Van Rossem	Jonathan Jeurissen
Configuration Manager	Jorne Laton	(Sina Khakbaz Heshmati)
Quality Assurance Manager	Patrick Provinciael	Bart Maes
Requirements Manager	Wouter Van Rossem	Jonathan Jeurissen
Design Manager	Bart Maes	Nick De Cooman
Implementation Manager	Sina Khakbaz Heshmati	Jorne Laton
Webmaster	Wouter Van Rossem	Sina Khakbaz Heshmati

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
5. Quality above quantity.

3.2 Assumptions, dependencies and constraints

- No assumptions are made.
- Dependencies
  For the project to succeed, it depends on the knowledge and the motivation of all team members. A team is only as strong as its weakest link.
- 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 manager.

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

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 manager.

3.3.2 Lack of knowledge of certain tools and mechanisms

Criticality: ++++
Some tools and mechanisms will be completely new for some team members. This can result in a lack of knowledge and inefficiency. To prevent this, tutorials will be made as wiki-pages and can be found at our Google Code website. Moreover, presentations are given if the subject is too complex or if it demands a complete explanation. During the implementation, we also organise workshops. These sessions allows members to ask advice and to help each other with encountered problems.

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 their likelihood, the impact on the project, the cost of disposal and their priority. This last one is calculated as follows :

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

The risk table is shown in table 3.1.

3.4 Monitoring and controlling mechanisms

3.4.1 Meetings

Meetings will be held regularly. The topics to discuss are defined in an agenda and will be sent to every team member before the start of the meeting. Minutes will summarize the meeting 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 three hours before the start of the meeting.

3.4.2 Timesheets

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

3.4.3 SCRUM Report

Every two weeks, all team members need to deliver a SCRUM report in which they briefly have to answer three questions:

1. What have you done in the last two weeks?
2. What will you do in the next two 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.

In reality however, this did not worked out. Since it was yet another administrative task, the support within the team was rather small. Therefore, the mechanism was stopped.

3.4.4 Highlights

Every month, a highlight report is published on our project website to give an overview of recent improvements. Being so, it allows the customer to monitor the project progress.

3.4.5 Development blog

The development blog allows to share our experiences related to the implementation. It collects our success stories and failues and describes thoughts about the tools we use. It is intended to be a communication channel during development in order to give an overview of our experiences.

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 (sometimes handy e.g. for meeting agenda).
  - File download server (for 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.
- **JBoss Seam**
  Seam is a powerful open source development platform for building rich Internet applications in Java. It enables developers to assemble complex web applications using simple annotated Java classes, a rich set of UI components, and very little XML. Seams unique support for conversations and declarative state management can introduce a more sophisticated user experience.
- **Eclipse/IntelliJ**
  Eclipse is a multi-language, cross-platform software development environment comprising an IDE and a plug-in system to extend it. It is used to develop applications in Java. Initially, it was decided to work with Eclipse because it has a very userfriendly interface and because of its popularity. It is also known by most team members.

  However, Eclipse was found too resource-demanding by some team member's, and therefore IntelliJ was chosen as an alternative IDE. Like Eclipse, IntelliJ is a cross-platform Java IDE, which offers support for the Seam Framework and the JBoss Application Server. All this makes it a perfect alternative for Eclipse.
- **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.

4.2 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 Statistics

Timesheets keep track of all our activities and provide a clear image about the work packages. As such, we can be very flexible in re-ordering responsibilities in order to have a well-balanced amount of working hours for each team member.

Please note that for Jonathan Jeurissen this course only counts for four study-points, while for the others it is weighted as nine.

5.1.1 Iteration 1

The following statistics provide a clear view on the work packages during the first iteration. This iteration started on October 12, 2009 and ended on April 04, 2010 and thus lasted 20 weeks in total. During this period, we started the project and made our project website, defined and described the requirements, chose and designed the system architecture and implemented the main functionality.

Since this iteration was spread over 20 weeks, we make a subdivision. Let us first analyze the timesheets statistics from the period October 12, 2009 - January 04, 2010 as given in table 5.1.

As 52 hours is a sufficient high weekly average, it indicates great motivation of the team. Despite this, the partition of worked hours was divided disproportionatly as shown by the pie chart in figure 5.1. Therefore, some resposibilities were re-organized to minimize the gap with Sina Khakbaz Heshmati on the one hand, and to have a clean difference between Jonathan Jeurissen and the others team members on the other hand. This re-organisation is treated in section 2.5.2.

Statistic	**Result**
Number of weeks	12
Total amount of hours	625
Average amount of hours/week	52

Table 5.1: General statistics for period 12/10/09 - 04/01/10.

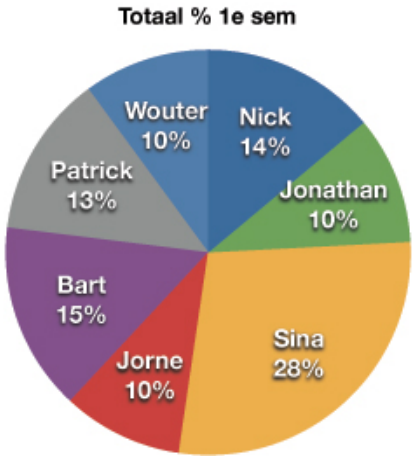

Figure 5.1: Percentual partition of working hours.
Period 12/10/09 - 04/01/10

Statistic	**Result**
Number of weeks	8
Total amount of hours	425
Average amount of hours/week	53

Table 5.2: General statistics for period 08/02/10 - 04/04/10.

Next, we analyze the statistics for the period February 08, 2010 - April 04, 2010. During this, everyone was encouraged to reach a specific target of working hours. This resulted in the statistics shown in table 5.2.

As we compare with table 5.1, we see that the weekly average remained more or less the same. However, figure 5.2 shows a substantial difference. The re-organisation of responsibilities led to a drastically change in the partition of working hours. This was obviously necessary in order to become a proportional working balance in the end.

In general, we present the statistics for the complete first iteration in table 5.3. The partition of working hours is prorated and well-balanced, taking into account the number of study-points. Every team members worked more or less the same amount of time. This is also shown by the pie chart in figure 5.3.

Statistic	**Result**
Number of weeks	20
Total amount of hours	1050
Average amount of hours/week	52.5

Table 5.3: General statistics for period 12/10/09 - 04/04/10.

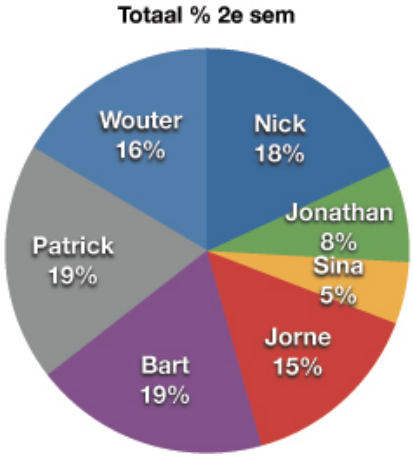

Figure 5.2: Percentual partition of working hours.
Period 08/02/10 - 04/04/10

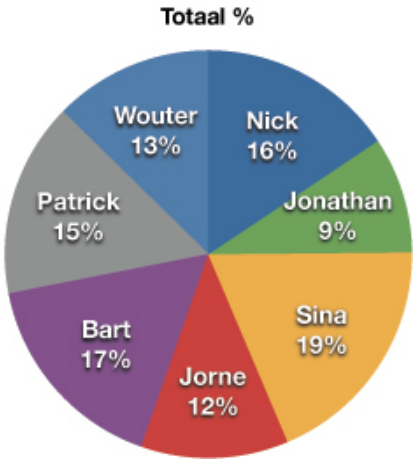

Figure 5.3: Percentual partition of working hours.
Period 12/10/09 - 04/04/10

5.2 Costs

5.2.1 Man-costs up to now

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

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

We know that the total amount of working hours, given in table 5.3, is 1050 hours. This allows us to calculate the current man-cost:

$$\text{Total man-cost} = 1.050 \text{ working hours} \cdot 14,50 \text{ wage/hour} = \text{€}15.225$$

5.2.2 Estimation of total man-costs

Based on the information in section 5.2.1, we can estimate the total cost of this project by using current weekly average of working hours. For the first iteration, this number is *52,5 hours/week*.

Since 27 weeks are foreseen to work on this project, the total number of man-hours currently can be estimated as:

$$52,5 \text{ hours/week} \cdot 27 \text{ week} \approx 1420 \text{ man-hours}$$

In that case, the estimation of total man-cost results in:

$$1420 \text{ man-hours} \cdot 14,50 \text{ euro/hour} = \text{€}20.590$$

Note that in previous document versions, the total man-costs was estimated more or less the same. Since the estimation is based on the weekly average of working hours, it indicates that this number is consistent over time.

5.2.3 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 5.4.

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

Parameter	Value
a	3.0
b	1.12
c	2.5
d	0.35

Table 5.4: COCOMO Parameters for a semi-detached project

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

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.

5.3 Dependencies

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

- Iteration 2 depends on the completion of iteration 1.

5.4 Planning

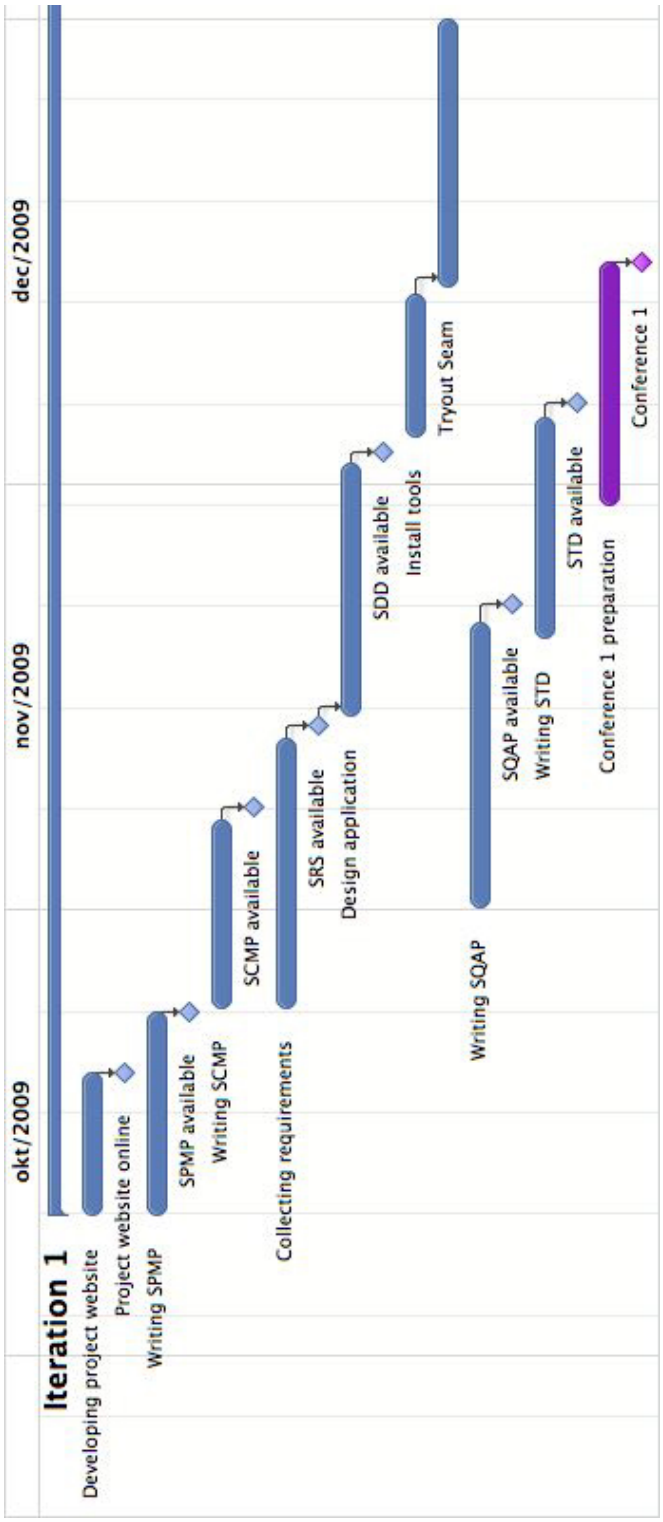

Figure 5.4: Planning iteration 1 in form of Gantt-chart - Part 1

Figure 5.5: Planning iteration 1 in form of Gantt-chart - Part 2

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.