

Software Project Management Plan

Software Engineering Group 5

Joke De Bock
jvdebock@vub.ac.be

May 15, 2003

Abstract

This document contains the *forum* Project's Software Project Management Plan (SPMP), which describes management of the production of a product within given time and funding limits. It documents the Project Plan so that every team member knows what to do, and when to do it.

This document is designed conform to the IEEE 1058.1-1987 standard for Software Project Management Plans.

Date	Author	Comment
2003-02-19	J. De Bock	Preview of first version
2003-02-22	J. De Bock	Completion of first version
2003-03-05	J. De Bock	Adaptations according to consultants remarks
2003-03-19	J. De Bock	Update of the SPMP according to the projects evolution
2003-04-06	J. De Bock	Update according to end of the first and beginning of the second iteration
2003-04-11	J. De Bock	Detailed work packages and weekly update
2003-04-20	J. De Bock	Weekly update
2003-04-27	J. De Bock	Update of workpackages
2003-05-04	J. De Bock	Update of workpackages
2003-05-05	J. De Bock	Update according to end of the second iteration
2003-05-07	J. De Bock	Update after in-progress audit
2003-05-11	J. De Bock	Weekly update
2003-05-15	J. De Bock	Conclusion of project

Table 1: Version History

1 Introduction

1.1 Project Overview

This project is organized to produce a system supporting a web-based *forum*.

This *forum* will be developed in several stages. We first built a prototype with only little functionality before beginning the implementation of a fully functional version of our *forum*.

1.2 Project Deliverables

The following are to be delivered at the times shown:

- Version 1 (prototype): March 20, 2003
- Version 2 (meeting all requirements): April 25, 2003
- Version 3 (with proviso): May 25, 2003

- Supporting documents:
 - SCMP: February 20, 2003
 - SRS: February 27, 2003
 - SDD: March 13, 2003 (CD due March 6, 2003)
 - SQAP (concerning Version 1): March 27, 2003
 - STD (concerning Version 1): March 27, 2003
 - SQAP (concerning Version 2): May 1, 2003
 - STD (concerning Version 2): May 1, 2003
 - SQAP (concerning Version 3): May 22, 2003
 - STD (concerning Version 3): May 22, 2003

Acronyms are defined in Section 1.5 below.

1.3 Evolution of the SPMP

This document shall be maintained on a weekly basis by the Team Manager. It is subject to configuration management by means of the SCMP. It is the Team Manager's responsibility to submit this document as a CI, and to keep it up to date.

1.4 Reference Materials

- [Braude] One of the principal sources of textbook reference material is "Software Engineering: an Object-Oriented Perspective" by E. Braude (Wiley, 2000)
- [Eckstein] UML pocket reference by R. Eckstein (O'Reilly & Associates, 1999)
- [Gamma] Design Patterns by E. Gamma, R. Helm, R. Johnson, J. Vlissides (Addison Wesley, 1995)

- [Oetiker et al.] The Not So Short Introduction to Latex2E by Oetiker et al. (1999)
- [Sommerville] One of the principal sources of textbook reference material is "Software Engineering 6th Edition" by I. Sommerville (Addison Wesley, 2001)
- [Stroustrup] The C++ Programming Language by B. Stroustrup (Addison Wesley, 1997)
- [Vermeir] Multi-Paradigm Programming using C++ by D. Vermeir (Springer-Verlag London, 2001)

1.5 Acronyms

- *CD*: Conceptual Design — the design that will be used to construct the prototype, it will be discussed internally but won't be described in a supporting document
- *CI*: Configuration Item — an item tracked by the configuration system.
- *CM*: Configuration Management — the process of maintaining the relevant versions of the project.
- *COCOMO*: Constructive Cost Model — cost estimation method based on the number of lines of code by Boehm.
- *QA*: Quality Assurance
- *USDP*: Unified Software Development Process — by Jacobson, Rumbaugh, Booch.
- *SCMP*: Software Configuration Management Plan — describes how changes to the project's documents will be handled.
- *SDD*: Software Design Document - describes the design of the project.
- *SPMP*: Software Project Management Plan — this document.
- *SQAP*: Software Quality Assurance Plan — identifies project documentation, standards, reviews, audits, and risk management.
- *SRS*: Software Requirements Specification — specifies customer and detailed requirements.
- *STD*: Software Test Document — describes the STP
- *STP*: Software Test Plan — the planning of unit, integration and system tests.
- *tbd*: to be decided

2 Project Organization

2.1 Process Model

This project is executed using a Spiral Development process with an iteration corresponding to each version.

This spiral process recognizes the need to visit the *requirements analysis / design / implementation / test* sequence more than once. Our main reason for this choice is the need to take care of the risks. Other reasons are that using this process model, we have built an early partial version which has been used to convince our customer of the quality of our eventual product, moreover it avoids integrating a large amount of code all at once.

The extent of the third iteration depends greatly on the progress of the first and second iteration. Given almost all requirements are met after the second iteration, we will make the existing version of the forum more robust and will add some requirements, after consultation of our customer.

The iterations are to be grouped according to the classification used in the USDP. The USDP groups iterations into Inception, Elaboration, Construction, and Transition iterations. The first iteration covers an Inception, Elaboration and Construction phase. The second iteration covers an Elaboration and Construction phase. Finally, the third iteration will cover an Inception phase followed by an Elaboration and Construction phase. It concludes with a Transition phase to complete the product release.

2.2 Organizational Structure

The project shall be organized as a team of peers with designated roles. The roles are *team manager*, *configuration manager*, *quality assurance leader*, *requirements management leader*, *design leader*, *implementation leader*, *database manager* and *webmaster*. Additionally, there are liaison roles to the customer.

These roles are shown in Figure 1.

Each team member will be included in the inspection of all work of the other team members.

Since each role is critical, we designated a kind of “buddy” for each leader. The buddy team member will keep in touch with all work concerning the role of the leader particularly well as in case the leader is incapacitated he/she will have to take over the responsibilities connected to this role.

2.3 Organizational Boundaries and Interfaces

The project team will interface with the following individuals representing the customer: Prof. D. Vermeir, Stijn Heymans.

2.4 Project Responsibilities

The responsibilities of the participants in the project are shown in Table 2.

Being responsible for a document includes the following:

- Making sure that the document is created on time
- Obtaining the team’s approval before publication

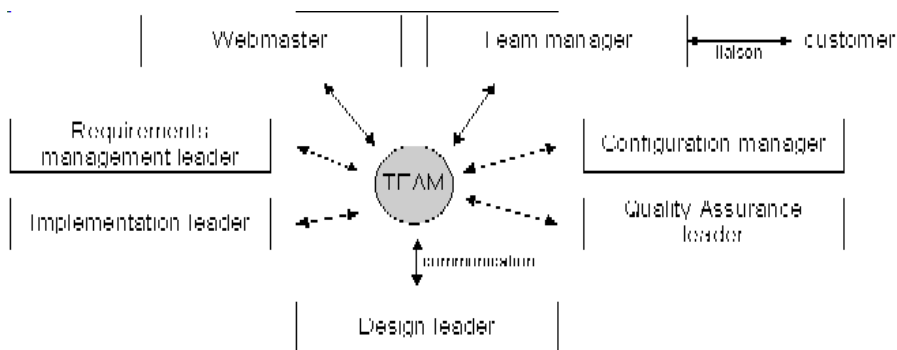


Figure 1: Project Organization

Member	Liaison Responsibility	Document Responsibility
Team Manager	Customer	SPMP
Configuration Manager		SCMP
QA Leader		SQAP, STD
Requirements Management Leader	Customer	SRS
Design Leader	Customer	SDD
Implementation Leader		Code Base
Webmaster		Website
Database Manager		Database

Table 2: Project Responsibilities

- Keeping the document up-to-date throughout the project life cycle

To obtain the team's approval one should add the request to review the document in the commit-message used in the CVS-repository. Additionally an email could be sent to the mailinglist se5. A document should be available for review at least one working day before it is finalized.

3 Managerial Process

3.1 Management Objectives and Priorities

The highest management priority shall be satisfying as many requirements as possible during the given time. The second priority is that the product is on schedule. This also includes the meeting of deadlines. The third priority of this project is to produce reusable classes for use by other projects. The fourth management priority will be the attainment of the specified quality parameters. For these quality parameters refer to the SQAP.

A desirable *forum* is expected only for version 2, the version obtained after the second iteration, and higher.

3.2 Assumptions, Dependencies and Constraints

As we will be using libraries that weren't written by one of the team members we will make the assumption these libraries either don't contain errors or are fixed by the original author in reasonable time. For the dv* libraries, support for existing functionality has been guaranteed by the author Dirk Vermeir.

Moreover we assume we are able to make the required changes to the XMLRPC-Call we would like to use. The problem is the original XMLRPC-Call assumes it is used with a URL while we will be using a DNS-address.

We assume MySQL supports full text search.

After thorough investigation we concluded the above assumptions about XMLRPC and MySQL are justified. These conclusions could already be drawn during the first iteration. This diminished the risk of relying on them while the assumptions didn't hold.

Some constraints could originate from the fact that all team members also have other projects to work on, however that shouldn't lead to problems when kept in mind.

3.3 Risk Management

Table 3 describes a format for risk reporting and retirement.

Each project meeting will have an agenda item to discuss the progress made concerning the retirement of existing risks and to tackle new risks arisen from the decisions made during the meeting, or not detected earlier. Team members are asked to report detected risks to the team manager who will foresee a retirement.

	R1: Hardware Failure	R2: Lack of Knowledge	R3: Waive of Function	R4: Lack of Communication	
Likelihood to Occur	0.2	2	1	3	
Impact	10	9	8	9	
Retirement Cost	1	6	5	1	
Priority Do in Order	10.8	108	150	16	
Retirement or Migration Plan	Backup Server	Tutorials and Presentations	Backup for Each Role	Communication meetings, icq, mail, irl	Week
Target Completion Date	2003-02-20	until end of project	2003-02-13	until end of project	unti

Table 3: Risk Table

Risk # 1 "Hardware failure" concerns the problem of personal and server hardware failure. In any case, we aim to keep the loss of data very low in case a hardware failure occurs. Therefore every team member will make every document or file containing code he/she has been working on publicly available through the CVS repository. In case of a personal hardware failure with data loss as a consequence, the concerning team member can get a copy of all work from the CVS repository. In case we'd have to cope with a server hardware failure, two problems might occur. As we wouldn't be able to check out or commit to detect changes, versions of the different team members might become

inconsistent. Moreover, the CVS repository might be damaged. Both cases will be taken care of by means of a backup server.

Risk # 2 "Lack of knowledge" concerns the problem of personal failure to understand a certain tool or technique we will be using. We tried to diminish the consequences of initial lack of knowledge by providing to the team members tutorials and presentations concerning some possible new techniques and tools each written by one of the team members.

This risk can be divided into a number of subrisks which will be described next.

- *Lack of SQL knowledge*
As we all had a course on databases covering SQL last year, the risk concerning the lack of knowledge about SQL should not be very large.
- *Lack of XML knowledge*
None of the team members had worked with XML before. The knowledge about XML could be seen as a risk given these circumstances. However, some of our team members already had a closer look on XML and XSLT and explained the use of it to the other team members.
- *Lack of C++ knowledge*
As we all had a course on Structure of computer languages, covering the C++ language, the risk concerning the lack of knowledge about C++ should not be very large. Moreover, all team members will be encouraged to work together to solve little issues and to improve the quality of each other's code and skills while implementing.
- *Lack of knowledge about Autotools*
As none of the team members has worked with autotools before, we assigned one of our team members to do some research concerning the possibilities to use it within our project. This way, the risk following from the possible lack of knowledge about Autotools has been diminished.
- *Lack of knowledge about Bugzilla*
None of the team members has used Bugzilla before. One of our team members has had a closer look on the possibilities to use it within our project and explained its use to the rest of the team.
- *Lack of knowledge about the overall situation of the project*
On the weekly meetings we try to give an overall view on the project. This way, every team member can situate his/her work into the whole project. This lessens the risk on misunderstandings between the several teams working on relating modules and increases the communication between the different teams.

Risk # 3 "Waive of function by a team member" concerns the human nature of the team members. It covers illness, giving up, ... Although this would make the whole project more complicated, a quite simple solution has been worked out. We will work with a buddy system, which means that for every function in the project there will be a backup. This buddy team member should keep in touch with all work concerning the function he/she might have to take over in such a way, the take over could be relatively smooth. Moreover, the team manager will have to understand every function of the project. This way she

can take over any function when that would be needed. To assure no-one loses touch with the whole picture, everyone will be asked to report and present his/her work during a meeting on a regularly basis.

Risk # 4 "Lack of communication within the team" concerns the communication within the group. The good termination of this project holds or falls with a good communication among the team members. Therefore it is very important to discuss the status of your work and your ideas and visions with other team members.

Risk # 5 "Meeting of deadlines" concerns the meeting of the deadlines, either internal or external. As every part of the work is dependent upon others, it is very important the work is completed by the end of the assigned time. Otherwise other, depending, pieces of work will be delayed as a consequence of the delay of its predecessors. If a task can't be completed given the assigned time, report the delay to the team manager and inform all team members working on dependent tasks. Every week during the meeting all deadlines will be evaluated and if needed, tasks can be rescheduled.

Risk # 6 "Quality not acquired" concerns the risk of not acquiring the quality, security procedures or the efficiency postulated. Our Quality Assurance leader is asked to perform tests concerning the quality and efficiency and to report any difficulties. Every team member is urged to keep the quality requirements, security procedures and efficiency requirements in mind and to report any problem detected.

Risk # 7 "Integration problems" concerns all problems that are detected during the integration. Integration may be the toughest part of the implementation process. All pieces have to be layed together and many problems can result from it. However, our team tries to keep the product working at all times. The latest stable version as well as our latest version possibly containing errors are available through the website. Every new functionality implemented is integrated in the product as soon as it has been finished. This way, a "big bang" has been avoided. Moreover, integration sessions are held whenever large pieces of new functionality had to be integrated.

Risk # 8 "Failure of mailinglist" concerns the risk of the failure of the mailinglist we use to communicate. Unfortunately, this risk hasn't been detected before the problem was detected. The mailinglist has been down for only one day. Although this isn't a real problem directly connected to the project, it has had a big impact on the communication within our team during the last two days. Of course, we can retire this risk by sending the emails to the emailaddresses separately when we notice our own email sent to the list doesn't arrive. Moreover, we have a backup mailinglist which can be used whenever the official mailing list fails.

Risk # 9 "Failure of tools" concerns the risk of the failure of the tools that are used throughout the project. We experienced tools as Bugzilla are not completely infallible. We always store a copy of the Bugzilla database, so that we can restore this database whenever needed.

3.4 Monitoring and Controlling Mechanism

The entire team will meet at the beginning of each phase (requirements, design, implementation and testing) of each iteration. There will be weekly project meetings on Thursdays from 01:00 P.M. to 03:00 P.M.. We will have to keep in

mind most of the team members can only stay until 03:00 P.M. because they have course at that moment. Though, every effort will be made to accomplish all team discussion during that meeting. Team members are asked to keep Tuesdayafternoon (01:00 P.M. until 03:00 P.M.) free for eventual additional meetings in case this would be necessary. The team manager will inform the team before Monday at noon when an additional meeting will take place. Of course, other team members can request an additional meeting by informing the team manager of this necessity.

In addition, various team members will be asked to report progress on a regularly basis. The team manager will have to report at every weekly meeting. She will be asked to write a project status form (Refer to Post Mortem Report) after each iteration. The configuration manager will be asked to report at every meeting. The quality assurance manager will be asked to report the quality of the project at every meeting. After each iteration he will have to write a post mortem report.

3.5 Staffing Plan

Table 4 shows how the different roles will be filled in.

Name	Proj. Mgr.	Conf. Mgr.	QA L.	RM L.	Design L.	Impl. L.	Webm.	DB Mgr.
Dieter S.		BU		BU	BU			
Joke D.B.	X		BU					
Joris B.	BU		X					
Michel M.				X				X
Niels J.		X				BU	X	
Peter V.						X		BU
Stijn M.					X			

Table 4: Staffing Plan

Our team will work within a horizontal organization structure because all team members are equals. Additionally, Joke De Bock is designated as the project manager. Ideally, she should encourage team members to participate, but she should also be decisive when necessary.

4 Technical Process

The SRS describes several aspects of the required technical process. This section provides information concerning aspects that aren't covered in the SRS.

4.1 Methods, Tools, and Techniques

The *forum* project is implemented in C++ using a linux-system. Doxygen has been used for documenting the code.

All documents supporting the project have been written in Latex.

MySQL is at the database-side of the project.

We made use of the advantages Autotools can offer for our project.

XML and XMLRPC are building stones of the project.
Bugzilla has been used for bug reporting.
See Section 2.1 for a description of the process.

4.2 Software Documentation

As stated in Section 4.1, we made use of doxygen for the documentation of the code.

The software resulting from our project is supported by a number of documents. Refer to Section 1.2 for a complete list of these documents.

4.3 Project Support Functions

Throughout the project the team has had the technical support of our consultant Dirk Vermeir. Dirk Van Deun provided us with technical support concerning the server the project will be running on, wilma, and the MySQL database.

5 Work Packages, Budget and Schedule

5.1 Work Packages

This section gives a very short description of the tasks that need to be completed and is based on our internal implementation plan written by our Implementation leader, Peter Vrancx.

5.1.1 Modules

Communication

This component provides communication between processes. It is an extension of the class XmlRpcCall. A parsing constructor is used in the receiver of the call. It can be used to get the string (methodName) and find a matching functional object for the call.

Database Component

This module provides database access. This component is derived from the DvMySQL Command Class. It parses queries and generates XML instead of vectors of rows. To do this an extra method to return XML is added to the existing inherited interface.

Client Component

Programs for the user. Its interfaces, webbased and command line, will be determined by the use of DvCGI and the forms discussed in the SRS.

Utilities Component

This component provides the means to perform pre- and postprocessing for every function. This will allow the external storage of certain user data. It is an essential component that allows the changing of certain appliance subsystems and thus enables more reuse.

Forum Component

This component consists of a series of functional objects that encapsulate the necessary forum functionality.

Interfaces

To enable efficient cooperation between implementation teams and integration of various project parts a series of interfaces needs to be defined. These interfaces are discussed and agreed upon by the entire team. Once determined an interface can only be changed with the consent of all team members. An interface for each module can be found in the design document.

5.1.2 General division of team members over specific modules

This section describes the general division of the team members over the different modules.

Independently of the module he/she is assigned to, every team member has been required to work on the project for at least one person-month spread over the whole project.

Division over the modules

- XSLT: Dieter
- Utilities: Stijn
- Client:
 - Dieter
 - Joke
 - Michel
- Forum:
 - Joris
 - Niels
 - Peter
 - Stijn

Buddies

Apart from the buddies assigned for each function within our team, we assigned buddies for the implementation process as well. Being a buddy includes the following:

- Assisting the team member you are the buddy of for the specific module
- Unit testing the module you are the buddy for

Within each module, the team members working on it are each others buddies.

Additionally, Niels has been Dieter's buddy for the work on the XSLT and Joke has been Stijn's buddy for the work on the Utilities module.

5.1.3 Work Packages

1. XSLT

- *Id:* 1.1
Team: Dieter, Niels
Description: update the XSL files to handle new use cases and improve the site lay-out
Internal deadline: Thursday April 17th
- *Id:* 1.2
Team: Dieter, Stijn
description: writing encoder for unrecognized chars
Internal deadline: Tuesday April 29th

2. Utilities

- *Id:* 2.1
Team: Stijn
Description: update existing use cases of the forum to use the Utilities Component; add functionality to retrieve a lost password
Internal deadline: Friday April 11th (12pm)
- *Id:* 2.2
Team: Stijn
Description: updated of the internal functional objects to the session Mangager to the rules and standards
Internal deadline: Thursday April 17th

3. Client

- *Id:* 3.1
Team: Joke
Description: implementation of the usecases:
 - Retrieving a lost Password
 - Editing a message
 - Uploading a File
 - Downloading a File
 - Editing user details*Estimated time needed:* 5u
Internal deadline: Thursday April 17th
- *Id:* 3.2
Team: Michel
Description: implementation of the usecases:
 - Sending a Mail
 - Closing a Topic
 - Copy Post to Category
 - Closing Category
 - Editing a Category

Internal deadline: Thursday April 17th

- *Id:* 3.3
Team: Dieter, Niels
Description: implementation of the command-line interface for the administrators of the forum
Internal deadline: Thursday April 17th
- *Id:* 3.4
Team: Dieter, Joke, Michel
Description: update existing use cases to conform with new agreements
Internal deadline: Tuesday April 22th
- *Id:* 3.5
Team: Michel
Description: implementation of the usecases:
 - Memberlist

4. Forum

- *Id:* 4.1
Team: Niels
Description: implementation of the usecases:
 - Sending a Mail*Internal deadline:* Thursday April 17th
- *Id:* 4.2
Team: Joris
Description: implementation of the usecases:
 - Requesting a new account
 - Requesting account details
 - Change account details
 - Editing a Category
 - Editing a Post*Internal deadline:* Thursday April 17th
- *Id:* 4.3
Team: Peter
Description: implementation of the usecases:
 - Closing Category
 - Copy Post to Category
 - Closing a Topic*Internal deadline:* Thursday April 17th
- *Id:* 4.4
Team: Peter, Stijn, Joris, Niels
Description: update existing use cases to conform with new agreements and database changes
Internal deadline: Tuesday April 22th
- *Id:* 4.5
Team: Stijn
Description: implementation of the usecases:

- Retrieving a lost Password
- Download a file
- Upload a File

Internal deadline: Thursday April 17th

- *Id:* 4.6
Team: Joris, Peter, Niels, Stijn
Description: (Adaptation of) Forum framework
Internal deadline: Tuesday April 29th
- *Id:* 4.7
Team: Peter
Description: implementation of the usecases:
 - Ban User
 - Memberlist

5. Database component

- *Id:* 5.1
Team: Every team member
Description: Reviewing the database component
Internal deadline: Tuesday April 22th
-

5.1.4 Dependencies

Iteration 2 is dependent on the completion of iteration 1 as we hope to be able to reuse some parts of the prototype built in iteration 1, and hope to learn from this iteration to make the second iteration (even) better than the first one.

As the first iteration ended, we noticed we will be able to reuse almost everything that has been written in the context of the first iteration for the further iterations.

The contents of iteration 3 will be dependent on the requirements met in the second iteration. In case all requirements are met in the second iteration we will be adding some features to our project in this last iteration. In case some requirements are not met yet, we will make every effort to complete the implementation of these requirements during the third iteration.

As the second iteration ended, almost all requirements are met. Consequently, we will work on these little issues during this third iteration. Moreover we will implement some additions, more specifically we will implement a memberlist, add ban-user functionality and make some lay-out changes to the product.

5.1.5 Resource Requirements

The project required seven engineers.

The minimal hardware resources required for this project are seven computers running Linux. The hardware resources consisted of each team member's personal computer running Linux, the computers located in the rooms at Infogroup and the Wilma-server.

5.2 Budget and Resource Allocations

5.2.1 Estimate before beginning requirements analysis

We have estimated the size of this effort using an informal top-down estimate based on previous experiences of the team's members.

Based on earlier experiences of various team members and the opinion of our consultant, the size of the effort is estimated to lie between 2000 and 3000 lines of C++ code.

Using the COCOMO cost model and given the assumption that we have to deal with a moderate project, the number of person months needed for a 2500-line project is 8.37. The estimated duration using COCOMO is then 5.25 person-months. In that case 1.6 engineers would be needed.

Although we have fewer time than that, we have more engineers assigned to our project. Given our 7 engineers and the need for 8.37 person-months it is easy to conclude that every team member will have to work 1.2 person months spread over the duration of the project. Furthermore, we assume a person-year, given a university degree in computer science, is worth 50K EUR.

Given these circumstances we would charge 38500 EUR for the project, of which 34897 EUR will be charged to pay the wages of the team members.

Note that these estimates are still very rough as requirements analysis hasn't been completed yet.

5.2.2 Estimate based on SRS and SDD

To be able to make an estimate based on the SRS and SDD without implementation, one can use function points to estimate the number of lines of code.

A first step to the estimate of the number of LOC is the identification of the functions. One should consider user-level functionality rather than programming-level functionality, this way the functions can be seen as a combination of program characteristics. Refer to table 8 for the identification of the functions.

This calculation gives us an estimate of 92 unadjusted function points.

A second step to the estimate of the number of LOC is the estimate of the general characteristics for function point. This can be done by estimating the significance (on a scale from 0 to 5) of a number of adjustment factors as can be found in the next table 6.

From this table we can calculate the adjustment factor to be used when calculating the adjusted function points: 1.09.

When we apply this adjustment factor to our unadjusted function points, we get a resulting value of 100,28 adjusted function points.

When we choose to take the minimum lines of code per function point for the language C++, which is 29, we get a result of 2908 lines of code for our *forum* project. This estimate approaches the estimate made before beginning the requirements analysis.

This 2900-lines-of-code project will require a 9.9 PM effort yielding a cost of 41190 EUR for the payment of the engineers wages and a total cost of 45500

Function	Usecases it represents	Function point contribution	estimated LOC
Search	Searching	7	203
Account administration	Requesting an account	17	493
	Approving an account		
	Changing account details		
	Requesting account details		
	Retrieving lost password		
	Logging into an account		
	Logging out an account		
Message-based functionality	Deleting a message	15	435
	Closing a topic		
	Posting a message		
	Reading a topic		
	Reading a message		
	Editing a message		
	Move post to a category		
	Remove category from a topic		
	Move topic to another category		
Category-based functionality	Making a new category	14	406
	Deleting a category		
	Closing a category		
	Editing a category		
	Browsing a category		
Ranking	Ranking a message	7	203
Uploading a file	Uploading a file	11	319
Sending an e-mail	Sending an e-mail	7	203
User-group-related functionality	Making a new user group	14	406
	Changing a user group		
	Assign user to a group		
	Delete user group		
	Setting group privileges		

Table 5: Identifying functions

EUR. It seems we have underestimated the costs before we started the requirements analysis.

5.2.3 Estimate at the end of iteration 1, before begin of iteration 2

After the first iteration, we completed 4 of the usecases and even more importantly we put up the framework for these and following usecases completing the work on the database module and the communication module.

Table ?? shows the actual lines of code needed to complete this iteration for the functions identified.

Interpolating this to the entire project, we get a new estimate for the size of the project. Of course we have to take into account the LOC of the framework. We needed 164 LOC to complete the communication module, we needed 55 LOC to complete the database module and needed 168 LOC for the first version of

Adjustment factor	Case study
Requires backup/recovery?	5
Data communications required?	5
Distributed processing functions?	2
Performance critical?	2
Run on existing heavily utilized environment?	4
Requires on-line data entry?	5
Multiple screens for input?	3
Master fields updated on-line?	3
Inputs, outputs, inquiries of files complex?	4
Internal processing complex?	4
Code designed for re-use?	4
Conversion and installation included?	0
Multiple installation in different orgs.?	0
Must facilitate change and ease-of-use by user?	3

Table 6: Adjustment factors

the XSLT.

This way, we get an estimate of 4283 LOC for the entire *forum* project. Again, it seems like we have underestimated the effort needed for the project as this would require 15 PM of work. With this amount of work, the project would cost 62500 EUR resulting into the double of our initial estimate.

Estimate at the end of iteration 2, before beginning iteration 3 The table below gives an overview of the actual lines of code for the usecases an estimate was made for:

The following table gives an overview of the different modules and the LOC needed to implement them.

This results into 12572 lines of code for the product resulting from the second iteration yielding the need of 51.1PM and a cost of 212576EUR for the payment of the engineers and a total cost of 250000EUR. Compared to the initial estimate, this seems almost impossible. Even the previous estimate gave rise to a 4283-lines-of-code estimate.

However, when the actual time spent on the product resulting from our second iteration is taken into account, everything seems more realistic. We spent 920 hours on this product, calculations are based on the timesheets up to week 32. This equals 7,89 person-months and corresponds to a cost of 32822EUR for the payment of the engineers and a total cost of 36500EUR. Given a 8,37PM initial estimate, the actual time worked on the product approaches the estimate.

Conclusion: Final calculation of cost The following table gives an overview of the different modules and the LOC needed to implement them.

This results into 17082 lines of code for the product resulting from the second iteration yielding the need of 72PM and a cost of 299640EUR for the payment

Function	Usecases it represents	estimated LOC	actual LOC
Search	Searching	203	NA
Account administration	Requesting an account	493	NA
	Approving an account		
	Changing account details		
	Requesting account details		
	Retrieving lost password		
	Logging into an account		
	Logging out an account		
Message-based functionality	Deleting a message	130	
	Closing a topic	150	
	Posting a message		167
	Reading a topic		163
	Reading a message		125
	Editing a message	140	
	Move post to a category	160	
	Remove category from a topic	150	
	Move topic to another category	170	
Category-based functionality	Making a new category	130	
	Deleting a category	160	
	Browsing a category		144
	Closing a category	130	
	Editing a category	150	
Ranking	Ranking a message	203	NA
Uploading a file	Uploading a file	319	NA
Sending an e-mail	Sending an e-mail	203	NA
User-group-related functionality	Making a new user group	406	NA
	Changing a user group		
	Assign user to a group		
	Delete user group		
	Setting group privileges		

Table 7: Identifying functions

of the engineers and a total cost of 300000EUR. Compared to the previous estimate the number of lines of code seems realistic even though the resulting Person Months are not.

However, when the actual time spent on the product resulting from our second iteration is taken into account, everything seems more realistic. We spent 1068 hours on this product. This equals 9,2 person-months and corresponds to a cost of 38272EUR for the payment of the engineers and a total cost of 40000EUR. Given a 8,37PM initial estimate, the actual time worked on the product approaches the estimate.

5.3 Schedule

The project's schedule is shown in Figure 5.3.

Function	Usecases it represents	estimated LOC	actual LOC
Search	Searching	203	174
Account administration	Requesting an account	493	247
	Changing account details		365
	Requesting account details		188
	Retrieving lost password		178
	Logging into an account		190
	Logging out an account		152
Message-based functionality	Deleting a message	130	238
	Closing a topic	150	172
	Posting a message		167
	Reading a topic		163
	Reading a message		125
	Editing a message	140	206
Category-based functionality	Move post to a category	160	193
	Making a new category	130	223
	Deleting a category	160	224
	Browsing a category		144
	Closing a category	130	173
	Editing a category	150	178
Ranking	Ranking a message	203	270
Uploading a file	Uploading a file	319	239
Sending an e-mail	Sending an e-mail	203	185
User-group-related functionality	Making a new user group	406	169
	Changing a user group		172
	Assign user to a group		84
	Delete user group		171

Table 8: Identifying functions

As shown in the picture, all external deadlines have been made on this final project day. We underestimated the work on the first iteration and its end was consequently delayed by one week. All tasks depending on the conclusion of this first iteration were also delayed. The second iteration started one week late. The STD has been delivered with delay.

As a consequence of this delay, we have delivered the second iteration 10 days later than planned. Given the 7-day delay of the first iteration, this can be justified. Despite this delay, the third iteration has been ended on time. We plan ended the implementation of the third iteration on May 13th, giving the team members who have to write concluding documents time to do so until May 15th. The project officially ends on May 25th.

Refer to the SQAP for the schedule of quality activities.

Module	LOC
Clients	5185
Communication	223
Database	96
Forum	4448
Utilities	550
XSLT	2060

Table 9: Version History

Module	LOC
Clients	6157
Communication	314
Database	129
Forum	7519
Utilities	673
XSLT	2290

Table 10: Version History

6 Additional Components

6.1 Index

Contents

1	Introduction	2
1.1	Project Overview	2
1.2	Project Deliverables	2
1.3	Evolution of the SPMP	2
1.4	Reference Materials	2
1.5	Acronyms	3
2	Project Organization	4
2.1	Process Model	4
2.2	Organizational Structure	4
2.3	Organizational Boundaries and Interfaces	4
2.4	Project Responsibilities	4
3	Managerial Process	5
3.1	Management Objectives and Priorities	5
3.2	Assumptions, Dependencies and Constraints	6
3.3	Risk Management	6
3.4	Monitoring and Controlling Mechanism	8
3.5	Staffing Plan	9
4	Technical Process	9
4.1	Methods, Tools, and Techniques	9

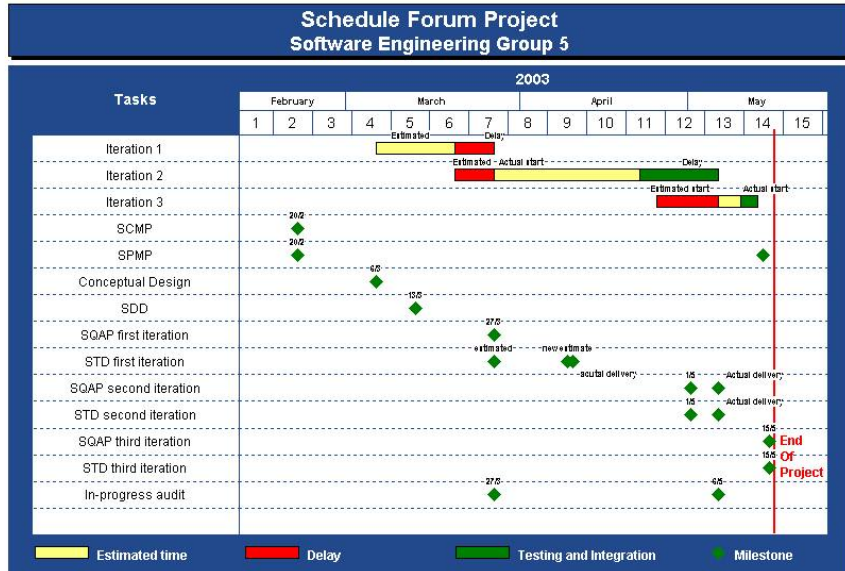


Figure 2: Schedule

4.2	Software Documentation	10
4.3	Project Support Functions	10
5	Work Packages, Budget and Schedule	10
5.1	Work Packages	10
5.1.1	Modules	10
5.1.2	General division of team members over specific modules .	11
5.1.3	Work Packages	12
5.1.4	Dependencies	14
5.1.5	Resource Requirements	14
5.2	Budget and Resource Allocations	15
5.2.1	Estimate before beginning requirements analysis	15
5.2.2	Estimate based on SRS and SDD	15
5.2.3	Estimate at the end of iteration 1, before begin of iteration 2	16
5.3	Schedule	18
6	Additional Components	20
6.1	Index	20
6.2	Appendices	20
6.2	Appendices	
	• Internal implementation plan written by Peter Vranx	