How to use this CD-ROM

You need

1) a WEB Browser. In most computers, a WEB Browser already exists. If you have any problem, contact us: cscc@worldses.org

2) Acrobat Reader (See the previous page of this cover).

FAQ (Frequently Asked Questions)

When I click on a link to a PDF file, nothing appears !

Probably your version of Internet Explorer, Netscape or Acrobat Reader is not recent enough, please try installing the ones provided in the CD-ROM. They are recent enough to work properly.

I have some trouble with printing (only one page printed, some additional text printed sometimes)

It s probably because you are printing from the Browser menu or icon bar. Try printing from Acrobat Reader icon bar. There is a small print button on Acrobat Reader icon bar which is located under the Browser icon bar.

How can I make zoom in the PDF files?

Click the appropriate button down and left or the buttons with the predefined standard zoom, up and right (below the Browser).

How can I make "copy" - "paste" in PDF files?

First Click the button with the letters abc, drag the mouse and then click at the "copy" button (up and left - just left from the button with the "palm"). After that, you can make "paste" in every Windows application.

How can I find something in a text?

In the HTML files, by pressing CONTROL+F. In the PDF files (Acrobat Reader), by pressing the button with the binoculars. You can also read the PDF files of this CD-ROM by using only the Acrobat Reader (without any Browser).

ISBN: 960-8052-19-X

ISBN of Vol.2 CSCC'99: 960-8052-01-7

4th World Multi-Conference on:
Circuits, Systems, Communications and Computers
(CSCC 2000)

2nd International Conference on:
Mathematics and Computers in Physics
(MCP 2000)

2nd International Conference on:
Mathematics and Computers in Mechanical Engineering
(MCME 2000)

IEEE

IMCS

STIR PALACE
OULIAGMENI
ATHENS
GREECE.
JULY 9-16
2000

4th World Multi-Conference on:
Circuits, Systems, Communications and Computers
(CSCC 2000)
(also containing Late Papers of CSCC'99)

2nd International Conference on:
Mathematics and Computers in Physics
(MCP 2000)

2nd International Conference on:
Mathematics and Computers in Mechanical Engineering
(MCME 2000)

ISBN: 960-8052-19-X
ISBN of Vol.2 CSCC'99: 960-8052-01-7

In order to access the documents a WEB Browser (Microsoft Internet Explorer or Netscape Communicator) and the Acrobat Reader software are required.
Some versions of the Acrobat Reader software are included for your convenience, which can be run on several platforms. In this CD-ROM the **Acrobat_Reader** directory contains the **latest versions (JUNE 17, 2000)** of the Acrobat Reader software for the following platforms.
1) Windows 95/98 2) Windows NT 3) Windows 3.1 4) Windows 2000 5) Solaris 6) Sun OS

Attention: Don't install the **Acrobat_Reader** , if you have already a newer version of it.
If you want to set up the ACROBAT READER program for other OPERATING SYSTEMS (Macintosh PPC, Macintosh 68K, IBM AIX, SGI IRIX, HP UX, Digital Unix, etc), please, visit http://www.adobe.com You can download the Installation programs from http://www.adobe.com/products/acrobat/readstep2.html
If there is any problem installing them, we advise you to contact Adobe Acrobat Inc. Ph: 1-900-555-2276, Mon-Fri: 6AM - 2PM (Pacific Time) or contact us: cscc@worldses.org
If you want to set up the ACROBAT READER program for other languages (for example Chinese, Arabic, French, Spanish, Portuguese, German, Italian, Japanese, Korean, Swedish, Hebrew), please, visit also http://www.adobe.com/products/acrobat/readstep2.html
Now, you can receive free weekly Adobe newsletters, plus the latest technical information, via e-mail. Please, visit: http://www.adobe.com/services/newsletter/subscribe.html and subscribe.

Make sure you install the Acrobat Reader with Search if you download from the Web Site. Search routine helps to locate any paper based on a keyword of your choice.

Techniques and Methods for Managing the Maintenance Process[1]

MACARIO POLO[+], MARIO PIATTINI[+], FRANCISCO RUIZ[+]
Alarcos Group
[+]Department of Computer Science
University of Castilla-La Mancha
Ronda de Calatrava, 7
13071-Ciudad Real (Spain)
{mpolo, mpiattin, fruiz}@inf-cr.uclm.es
*Organización y Estructura de la Información
Madrid Technical University
Ctra. de Valencia, Km. 7.
28031-Madrid (Spain)
jgs@eui.upm.es
http://alarcos.inf-cr.uclm.es

Abstract: - In this paper, a methodology focused on the improvement of the software maintenance process management is presented. As it is well-known, this one is the most costly stage and less planneable stage of the software life cycle. This methodology approaches maintenance as an horizontal set of activities and tasks (in the sense that they have an extension along the time) and proposes, for each task, the use of certain techniques, which can have been extracted from literature or have been propounded as additional work lines of this research. This full set of solutions is being used by a multinational organization which provides software maintenance services to third-party software systems.

Key-Words: - Software maintenance, maintenance techniques, maintenance management, life cycle.

1 Introduction

Spite software maintenance absorbs most the effort of the software life cycle ([4], [15]), most organizations do not follow any particular methodology for software maintenance, although they have one for software development [14]. As evidenced by recent studies, programmers spend most their time in maintenance tasks that developing new systems ([24]). To do maintenance works without a methodological basis is one of the main problems for software evolution, since programs get more complex after interventions and, consequently, less understandable and modifiable ([22], [7], [1]).

Moreover, novel environments and technologies will require great maintenance efforts to keep them operatives: Brereton et al. [2], for example, affirm that the maintenance of hypertext documents will probably become a serious problem which requires immediate action. For Hanna, as more software production, more maintenance effort [8], and the first one has always shown a growing tendency. Related with this, Lehman et al. have confirmed one of his laws (big programs never are completed and they are always in constant evolution [12]) almost twenty years after its announcement [13].

With this situation, organizations need complete guides for software maintenance which help *programmers* to carry out adequately this process, in such a manner that *managers* can keep the process under their control. This kind of guides (or methodological solutions) are also very interesting for the possible *customers* of software services organizations (like those which provide software maintenance), since they may help to guarantee the service quality. If such methodologies are based upon more global frameworks, the offered service is a good added value which will be well valued. Outsourcing of software life cycle activities is a growing business area in many sectors influenced by Information Technologies [21].

This article presents a methodological approach for carrying out the maintenance process., and its is organized as follows: in section 2 we present the current version of MANTEMA, a methodology for software maintenance, explaining in subsections some of its characteristics and techniques. Section 3 is an exposition of our conclusions and future lines of work

2 Structure of the Maintenance Process

The maintenance process (as well as the full software life cycle) may be understood as an *horizontal* set of tasks which is extended along the time. In some moments of the process, certain *vertical* techniques can be used for executing the respective tasks. Figure 1 illustrates this idea:

[1] This work is part of the projects: *MANTIS* (CICYT 1FD-097-1608) and *MPM: Mejora del Proceso de Mantenimiento,* developed with Atos-ODS (ATYCA, Ministerio de Industria y Energía TA15/1999).

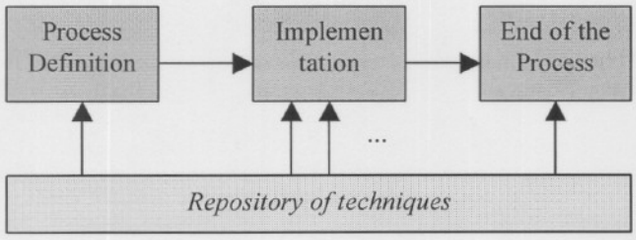

Figure 1. The horizontal structure of

The first node in Figure 1 is dedicated to prepare the maintenance process, whereas the second one is triggered when a modification request arrives to the maintenance organization to be served. In the third node, tasks needed for ending the process are executed.
Every modification request involves a change on the software which, in our proposal, must be categorized into one of the following types [17]:

a) *Urgent corrective*: a detected error prevents normal system operation and the solution time is critical.
b) *Non-urgent corrective*: a detected error does not block the normal operation of the system and the solution time is not critical.
c) *Perfective*: when new functionalities are added to the system.
d) *Preventive*: consists of the software modification to improve its maintainability and quality properties.
e) *Adaptive*: when the system will change its execution environment.

Following recommendations of different references ([9], [20], [10]), different set of tasks must be executed depending on the maintenance type of the modification request. In our methodology, a different set of tasks is defined for each type of maintenance.

2.1 Structure of a task

Once the modification request has been categorized into one type, it is very important to submit it to well-defined tasks, in the sense of providing all the information which is needed for its successful execution. Figure 2 illustrates the set of information that we use to define a task:

a) Input elements.
b) Outputs to other task or to the environment.
c) Interfaces with other processes of the software life cycle (Configuration Management, for example).
d) People in charge of the task.
e) Techniques which may help to the satisfactory execution of the task.
f) Metrics which must be collected to measure and evaluate different kinds of properties of our process.

2.1.1 Input and output elements

With the term "Input and output elements" we denote any possible software product which can be manipulated during the execution of a task.

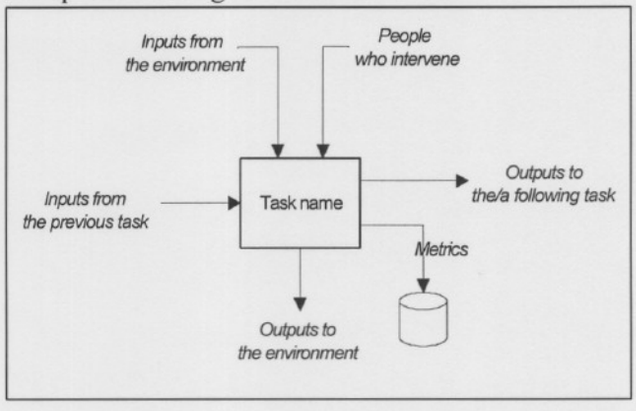

Figure 2. Generic structure of a task.

Input elements may come so from a previous task (an authorized urgent-corrective modification request, for example) as well as the from environment (the program with errors referred in such modification request). In the same manner, output elements will go directed to a following task or to the environment: the first case could be the modification request, when it has been served and will be registered; the second case would be the fixed software product which is being put into the production environment.
Therefore, both kind of "elements" will be generally constituted by "logical software products" (programs, databases) and documents. The methodology defines templates for all the possible documents suitable of being generated during maintenance. As an example, Figure 3 shows the template for a maintenance request.

2.1.2 Interfaces with other processes

In [16], we have shown how to apply the Tailoring Process defined in ISO/IEC 12207 [10] to the Maintenance Process defined in this same standard for defining our methodology.
As all processes in this standard, the Tailoring is defined as a set of activities and tasks. Very briefly:

1) Identifying project environment, whose only task consists of identifying those project environment characteristics which will influence on the tailoring (e.g., the life cycle model, system and soft-

ware requirements, size, criticality and system types, software product or service, etc.).

2) Soliciting inputs, whose only task consists in the solicitation of the inputs of the organizations which will be affected by the tailoring decisions (e.g., users, support personnel, etc.).

Modification request.
1) Solicitant identification
  1.1 Name
  1.2 Departament
  1.4 Phone number
2) Product identification
  2.1 Project
  2.2 Component and version
3) For modification requests by error
  3.1. Description of the application which fails and of its function
  3.2. Circumstances of the error (date and time, real input and output data, hoped input and putput data, free text)
  3.3. Error messages given by the system
  3.4. Recommended solution (if it is possible)
  3.5. Urgency degree and hoped date of solution
  3.6. Free text
4) For modification requests by functionalities addition
  4.1. Description of the desired functionality
  4.2. Justification of the addition
  4.3. Free text
5) For modification requests of software quality
  5.1. Description of the properties that can be improved
  5.2. Justification of the proposed improvement
  5.3. Free text
6) Place, date and time of the request presentation

Figure 3. Modification request template.

3) Selecting processes, activities and tasks, which (without underestimating the importance of the others) is the main task of Tailoring. This activity consists of three tasks:
   3.1. Decision about what processes, activities and tasks will be performed, including the documentation to be developed and responsible.
   3.2. Provision of the processes, activities and tasks selected in the previous task and not provided by ISO/IEC 12207.
   3.3 Paying special attention to those processes, activities and tasks which will be deleted and which are considered as requirements in the standard.
4) Documenting and rationalizing together all tailoring decisions.

Obviously, for defining a maintenance methodology, the most important process of those defined in ISO/IEC 12207 is the Maintenance Process. From this one we have obtained the "macrostructure" which was shown in Figure 2. However, several processes of the international standard provide different characteristics which can be useful during maintenance.

In order to use these characteristics, several of the processes defined in ISO/IEC 12207 have been integrated into the maintenance process defined by the methodology, whereas some others are "called" in determined moments of the process. Thereinafter, the Audit and Improvement processes leave aside, neither integrated nor interfaced. After these considerations, our maintenance process can be seen as the following "arachnid" graph, which shows, as satellite processes, those which are interfaced by maintenance:

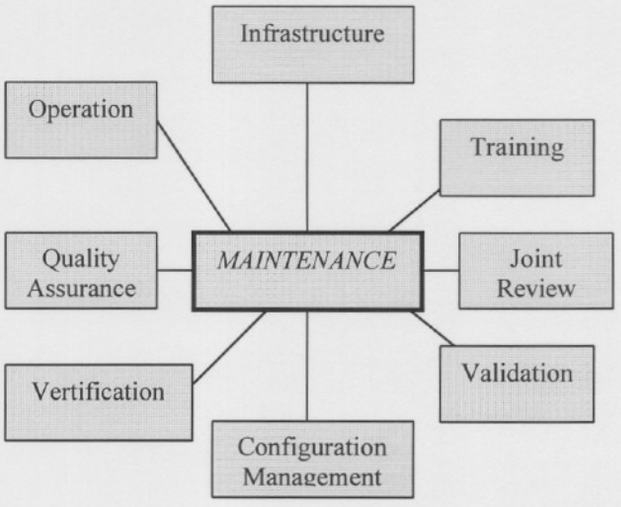

Figure 4. Life-cycle processes interfaced by Maintenance

2.1.3 Responsible people

Traditionally, most methods and techniques developed to support software processes have been built without taking into account organizational structures [6]. In the concrete case of software maintenance, very little has been written about the modeling of maintenance organizations [15].

In this methodology, a response to this lack of proposals is given. In fact, we define the following organizations and profiles [18]:

- Customer organization.

This is the organization which owns the software and requires the maintenance service. It has the following profiles:

1) The Petitioner: who promotes a Modification Request and establishes the needed requirements for its implementation and informs to the maintainer.
2) The System organization: this is the department that has a good knowledge of the system that will be maintained. This profile is useful because
3) The Help-desk: this is the department which attends to users. It also reports to the *Petitioner*

the incidents sent by *users* to generate the Modification Request.

- Maintainer profiles.

This is the organization which supplies the maintenance service.

1) The Maintenance-request manager: decides whether the modification requests are accepted or rejected and what type of maintenance should be applied. He/She gives every Modification Request to the *Scheduler*.
2) The Scheduler: must plan the queue of accepted modification requests.
3) The Maintenance team: is the group of people who implement the accepted modification request. They take modification requests from the queue, which is managed by the Scheduler. The structure of this team is an affair that must be sorted out by the Maintainer. An organization which provides software maintenance services according to the methodology must have identified these profiles (excepting the possible tailoring cases mentioned in section 2.1.1), but the internal structure of the team is not responsibility of the methodology.
4) The Head of Maintenance: prepares the maintenance stage. He/She also establishes the standards and procedures to be followed with the maintenance methodology used. He/She plays an important role in the initial set of activities of the methodology.

- User profile.

The organization that uses the maintained software.

1) The User: makes use of the maintained software. He/She communicates the incidents to the Help-desk.

Obviously, these ones are organizations from a "logical" point of view, since two or even three "physical" and real organizations may coincide in just one (for example, an enterprise which owns, uses and maintains its software). In the same manner, inside each organization, different profiles may coincide in only one person.

2.1.4 Techniques for Maintenance

In our context, a technique is a method which can be used for executing a maintenance task. Most techniques for maintenance proposed in the last years are focused on reengineering, reverse engineering and restructuration of programs and databases, having many of them the support of specific automatic tools. This kind of techniques is very suitable of being used in the Implementation node of Figure 1. For our initial node, the techniques we can found in literature are related with the effort estimation of future maintenance interventions, proposals for preparing outsourcing maintenance contracts, identifying risks and so on. In the methodology we propose to use a technique to know the quantity of resources which must be devoted to non-planneable maintenance with no economical loss.

2.1.4.1. Planneation of resources.

This problem can be studied from different points of view, depending on the existence/inexistence of outsourcing relationship and on which organization (of those listed in section 2.1.3) is doing the calculus.

In any case, all we need to apply this method is:

1) An estimation of the future distribution of modification requests (see, for example, [11], [23], [3]).
2) Some values of the Service Level indicators.
3) Some values of the following economical parameters of the project:
   3.1. Cost (from the Customer's point of view) of every contracted hour.
   3.2. Cost (from the MO's point of view) of every dedicated hour of resource.
   3.3. Cost of every system stop (fixed and variable).
   3.4. Sanctions imposed by the Customer to the MO for incurring in delays (fixed and variable).

When we have these values, the situation may be studied from these points of view:

1) There is outsourcing:
   - The MO may prefer to incur in delays if the sanctions plus costs of devoting less resources than needed is less than the cost of devoting all the needed resources.
   - The Customer will prefer that the MO delays if sanctions paid by the MO are greater than the loss plus the cost of resources
2) There is no outsourcing (the Customer, which is also the Maintenance Organization, maintains its own software):

   In this case, there are no sanctions and there is only a cost of resources. This organization will be interested in dedicating less hours of resource than the needed hours to fix all the errors from the point where costs of devoted resources plus loss by errors are equal to costs of needed resources to fix all the predicted errors.

In [19], we can find a mathematical model of this technique. Each organization has a cost and a benefit function. Figure 5 (see at the end of this paper) shows one example of the benefit function for a fictitious project with outsourcing and two organiza-

tions, depicted as functions of the planned hours of resource (p).

The behavior of the model is not so good far from the cutting point of both functions: in spite of the example has shown that the Customer would obtain great profits if the Maintenance organization devotes very little resources (because the MO would pay a lot of sanctions), the reality is that very big delays may imply unacceptable production losses. This means that, in this cases, other costs appear, not directly considered in this model. A similar effect occurs from the Maintenance organization's point of view, since the little dedication may imply others high risks and costs, as the cancellation of the maintenance contract. This fact may be included in the model considering, as an additional cost, a percentage of the losses suffered by the Customer, well linearly or exponentially (see [19] for more details).

2.1.4.2. Risks identification and estimation.

Before the acceptation or (if it is mandatory) execution of a maintenance project, it would be very useful to do an estimation of its most important risks.

There are some methods for managing risks of maintenance projects, but they are mainly focused on risks of concrete interventions. In MANTEMA, a technique for quantifying risks of the project when there is very little objective and quantifiable information about it has been developed. We have taking some situational factors from Euromethod [5] and from the Atos ODS experience.

Euromethod defines a situational factor as "a property which generates risks". Therefore, every situational has influence on certain types of risks. The types of risks we have identified are the followings:

1) Uncertain or unfeasible requirements
2) Uncertain interfaces to other systems
3) Evolving requirements
4) Unpredictable costs for the supplier organization
5) Unpredictable costs for the project
6) Delays in the delivery
7) Poor quality of deliverables
8) Increased costs of the project
9) Integration problems
10) Straining computer science capabilities
11) Wrong or unfeasible information system
12) IS-adaptation not accepted by actors
13) Business implications of project failure

The magnitude of these risks is estimated as a function of some of such situational factors. A subjective value is given to every situational factor according to the user of this method feels absolutely agree to absolutely disagree (respectively 1 to 5) with the sentence used to explain the factor. We have identified 36 situational factors, related to these aspects:

Factor	Subfactor	Value	Mean value
Existing specifications in the application	Quality of documentation is high	3	3,67
	Quality of system architecture design is high	4	
	Quality of software design is high	4	
	Documentation is updated	5	
	IS requirements are available and clear	3	
	IS requirements are stable	3	

Figure 6. An example of the table used for estimating situational factors

1) Quality of the system (documentation, system architecture, software design, documentation updating).
2) IS requirements (availability, clarity, stability).
3) Characteristics of the current maintenance team
4) Organization dependencies (extern changes, subcontractors)
5) Business processes (complexity, stability)
6) Information stability
7) Normalization level of the system
8) Replications of the IS is low
9) System criticality
10) Interfaces with other IS applications
11) Impact of errors on the organization public image
12) Critical moments (randomly, periodically)
13) Software vulnerability to user's actions (direct updates on the database, for example)
14) User dependence to know the problem
15) Denomination and programming standards
16) Quality of programs (batch and on-line)
17) Influence on reports and screens of maintenance interventions
18) Environment stability
19) Dependence of organizational changes
20) Change rate in business processes
21) Availability of technology used in the project
22) Use of methodologies during the system development
23) Use of tools during the system development

Figure 6 shows a filled-in fragment of the table used to estimate situational factors, whereas Figure 7 is a piece of the table used to estimate risks as a function of some situational factors.

3 Conclusions and future work

We believe that the version of MANTEMA briefly presented here is a full methodology for supporting maintenance, which may be very useful for software

organizations in general, but specially for those devoted to provide outsourcing maintenance services.

As we have noted in section 2.1.2, the Audit and the Improvement processes of ISO/IEC 12207 are not used in any moment of our maintenance process. Now, we are working in these two fields: control goals for auditing maintenance and, more recently, process improvement with related frames, like Spice.

References:

[1] Baxter, I.D. and Pidgeon, W.D. (1997). *Software Change Through Design Maintenance*, Proc. of the ICSM, 250-259. IEEE CS, Los Alamitos, CA.

[2] Brereton, P., Budgen, D. & Hamilton, G. (1999). *Hypertext: the Next Maintenance Mountain*. Computer, 31(12), 49-55.

[3] Calzolari, F., Tonella, P. and Antoniol, G. (1998). *Modelling Maintenance Effort by Means of Dynamic Systems*. Proceedings of the 3rd European Conference on Soft. Maint. and Reengineering. Amsterdam (The Netherlands), IEEE CS, Los Alamitos, CA, USA.

[4] Card, D.N. y Glass, R.L. (1990). *Measuring Software Design Quality*. USA: Englewood Cliffs.

[5] Euromethod version 1. Euromethod Project, 1996.

[6] Fuggetta, A. (1999). *Rethinking the models of software engineering research*. The Journal of Systems and Software, (47), 133-138.

[7] Griswold, W.G. and Notkin, D. (1993). *Automated Assistance for Program Restructuring*. ACM Transactions

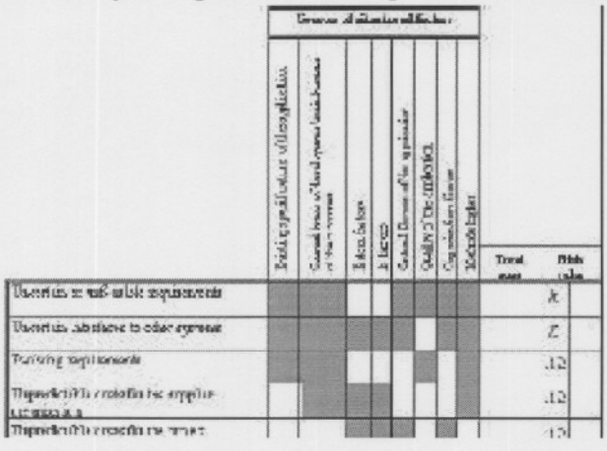

Figure 7. Table for estimating risks.

on Soft. Engineering and Methodology, 2(3), 228-269.

[8] Hanna, M. (1993, april). *Maintenance Burden Begging for a Remedy*. Datamation, pp. 53-63.

[9] IEEE Std. 1219-1992 (1992). Standard for Software Maintenance

[10] ISO-International Standard Organization (1995). ISO/IEC 12207. Information Technology Software Life Cycle Processes..

[11] JØrgensen, M. (1995). *Experience with the Accuracy of Software Maintenance Task Effort Prediction Models*. IEEE Trans. on Soft. Eng. 21(8), 674-681.

[12] Lehman, M, M. (1980). *Programs, Life Cycles and Laws of Software Evolution*. Proceedings of the IEEE, 19, 1060-1076.

[13] Lehman, M.M., Perry, D.E. and Ramil, J.F. (1998). *Implications of Evolution Metrics on Software Maintenance*. Proc. of the Int. Conf. on Software Maintenance. IEEE Computer Society, USA.

[14] Piattini, M., Villalba, J., Ruiz, F., Fernández, I., Polo, M., Bastanchury, T. y Martínez, M.A. (1998a). *Mantenimiento del software: conceptos, métodos, herramientas y outsourcing*. Madrid: Ra-Ma.

[15] Pigoski, T. M. (1997). *Practical Software Maintenance. Best Practices for Managing Your Investment*. John Wiley & Sons, USA.

[16] Polo, M., Piattini, M., Ruiz, F. and Calero, C. (1999). *Using the ISO/IEC 12207 Tailoring Process for Defining a Maintenance Process*. Proc. of the 1st IEEE Conf. on Stand. and Innov. in IT (SIIT '99). Germany.

[17] Polo, M., Piattini, M., Ruiz, F. and Calero, C. (1999). *MANTEMA: a complete rigorous methodology for supporting maintenance based on the ISO/IEC 12207 Standard*. Proc. of the 3rd European Conf. on Soft. Maint. and Reeng. IEEE CS, Los Alamitos, California (USA).

[18] Polo, M., Piattini, M., Ruiz, F and Calero, C. (1999). *Roles in the Maintenance Process*. Software Engineering Notes, 24(4), 84-86.

[19] Polo, M., Piattini, M. and Ruiz, F. (2000). *Planning the non-planneable maintenance*. Accepted in ESCOM-SCOPE 2000 (Munich, april).

[20] Pressman, Roger S. (1993). *Software Engineering: a practitioner's approach* (Third edition). McGraw-Hill.

[21] Rao, H.R., Nam, K. and Chaudhury, A. (1996). *Information Systems Outsourcing*. Communications of the ACM, 39(7), 27-28.

[22] Schneidewind, N.F. (1987). *The State of Software Maintenance*. IEEE Trans. on Soft. Eng., 13(3), 303-310.

[23] Shepper, M., Schofield, C. and Kitchenham, B. (1996). *Effort estimation using analogy*. Proc. of the 18th Int. Conf. on Soft. Maint. IEEE CSP, Los Alamitos CA.

[24] Singer, J. (1998). *Practices of Software Maintenance*. Proc. of the ICSM.

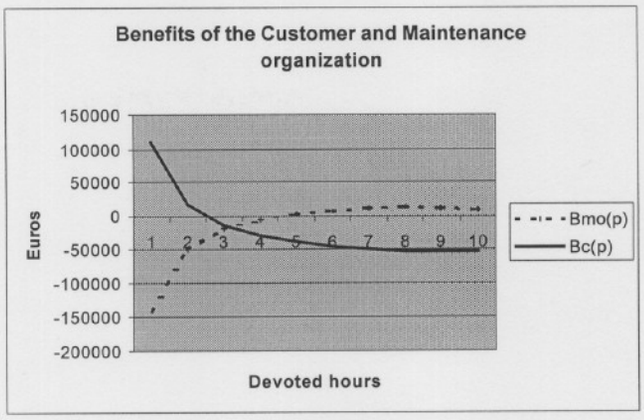

Figure 5. Benefits of the Customer (Bc(p)) and the Maintenance Organization (Bmo(p)), in an outsourcing project.