Roles In The Maintenance Process[1]

Macario Polo, Mario Piattini, Francisco Ruiz, Coral Calero
Grupo ALARCOS; Escuela Superior de Informática
Universidad de Castilla-La Mancha.
E13071-Ciudad Real (Spain)
{mpolo, mpiattin, fruiz, ccalero}@inf-cr.uclm.es

Abstract

Software maintenance is the most expensive and least predictable stage of the software life cycle, representing in some cases between 67% and 90% of total costs.

On the other hand, it is well known that the structure of a work team influences on the productivity of its members. We can help to decrease costs of maintenance through the definition of an adequate structure of the work team and with the clear identification of the tasks which every member must execute.

In this work we expose the results that, in this sense, we have obtained from the application of MANTEMA to real projects. MANTEMA is a methodology for supporting software maintenance, developed jointly by our university and Atos ODS, an international enterprise among whose primary business activities is the outsourcing of software maintenance. Atos ODS is using MANTEMA in big projects of Spanish banking.

Keywords: Maintenance, Software Life Cycle, Work team.

Introduction

Costs of maintenance, which is the most costly stage of the software life cycle [1], are influenced by a big set of factors, well characterised in the literature. Among they [2] has pointed out the following as the most important ones:

- Modification requests study
- Documentation study
- Source code study
- Source code modification
- Documentation updating
- Test design and execution

On the other hand, it is known that the structure and other properties of work team have some influence on their productivity [3]. So, for example, the making-up of teams with no more than ten persons is recommended [4] [5], in order to obtain a high level of cohesion among its members and to avoid an excessive number of communication views, which would exert a bad influence on team productivity.

Setting aside these qualitative appreciations, there are not, however, many quantitative studies which reflect the impact of the properties of work teams on life cycle costs (except some versions of Brook's Law [6]).

In the Maintenance process, the possibility of heading off any source of costs must be seriously analysed and studied. In this paper, we discuss how to help to reduce costs of Maintenance, through the definition of the different organisations which must intervene in this process, and providing a set of well-defined profiles for every one.

These results are the product of the application of MANTEMA methodology for software maintenance [7] to Spanish banking projects.

Our environment and Work method.

MANTEMA has been developed in the University of Castilla-La Mancha, in collaboration with Atos ODS, an international organisation among whose main business activities is the outsourcing of software maintenance.

Outsourcing is a fast growing activity, with an increasing number of organisations providing this service [8]. Then, in this highly competitive market, the implementation of a strict methodology for supporting maintenance of third-party organisations (besides it facilitates the sale of this kind of service) turns the Maintenance into a well defined, controlled and measurable process.

The outsourcing of software maintenance involves at least three organisations -all of them with some common goals (the right operation of the software, for example), but also with some opposite interests (money)- which we will discuss later:

- The Maintainer
- The Customer
- The User

The right definition of the relationships among these three kinds of organisations was a very important issue during the building of MANTEMA. Bringing together, incorporating relationships has been done at "task level": briefly, in MANTEMA we define five sets of activities to be performed for each of the five types of maintenance interventions we distinguish (urgent corrective, non-urgent corrective, perfective, preventive and adaptive). Furthermore, two special sets of activities (common to all intervention types) are defined to be carried out when the maintenance process is going to be contracted out and when this is going to be finished (after all the maintenance interventions).

[1] This work is part of the MÁNTICA project (CICYT 1FD-097-0168) and the MANTEMA project (carried out by Atos ODS, S.A. and Universidad de Castilla-La Mancha; ATYCA, Dirección General de Tecnología y Seguridad Industrial of the MINER, Ministerio de Industria y Energía, Spain).

Following the idea of the ISO/IEC 12207 International Standard [9], that we have taken as a basis for constructing the methodology, every activity is composed of a set of tasks. For every task, we define the required inputs in order to execute the task according to some techniques we also describe, what outputs must be produced for executing the following task, what metrics must be collected and -we already arrive at the main purpose of this paper- who are the people who must take part in the task.

In the first stages of MANTEMA creation, we did not define strictly what people should take action in every task. Obviously, we knew that the modification of a line of source code could not be done either by an user or by the head of the customer organisation. However, we considered that the nature of every task itself was meaningful enough to ensure that every person knew what tasks he/she should take part in.

But the valuable feedback proceeding of the practical application of the methodology, which has followed the Action-Research method [10], revealed some little communication problems in certain tasks, especially of those belonging to the common initial and final activities and tasks, and of those executed at the beginning and at the end of the maintenance interventions.

Definition of people.

Taking as a starting point the three roles mentioned in the previous section, and knowing that a distinction must be made for distinguishing the personnel, we define some profiles for every organisation.

This definition is based on [11], [12] and our own practical experience acquired in the application of the methodology.

Before detailing and explaining these profiles, we define the three organisations as follows:

- <u>Customer organization.</u>

This organisation corresponds with the Acquirer defined in ISO/IEC 12207 [9]. We define it as the organisation which owns the software and requires the maintenance service.

- <u>Maintainer</u>.

The organization which supplies the maintenance service.

- <u>User</u>.

The organization that uses the maintained software.

The profiles we define for every one are listed below. This enumeration is not rigid, since it may be tailored to every particular case, as we will see in the next section.

- <u>Customer profiles</u>.

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

- <u>Maintainer profiles</u>.

1) The <u>Maintenance-request manager</u>: 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 <u>Scheduler</u>: must plan the queue of accepted modification requests.
3) The <u>Maintenance team</u>: 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 (and, in general, of any possible MANTEMA team) is an affair that must be sorted out by the Maintainer. An organisation which provides software maintenance services according to MANTEMA must have identified these profiles (excepting the possible tailoring cases mentioned in section 4), but the internal structure of the team is not MANTEMA's responsibility.
4) The <u>Head of Maintenance</u>: 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 common initial set of activities of MANTEMA methodology.

- <u>User profile</u>.

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

Tailoring.

The activities and tasks structure of MANTEMA can be tailored to different types of maintenance processes, with or without outsourcing, for example.

In this section, we focus our attention in the tailoring of MANTEMA participant organisations and their profiles. The main idea to do this tailoring is to assume that different profiles can coincide with the same person or group of people. Therefore, when there is just one organisation which uses, owns and maintains the software product, the three afore-mentioned organisations coincide with the "super-organisation" and it is the "super-organisation's" responsibility to gather the different organisations in itself, and even, of concentrating different organisations or profiles in only one person.

In conclusion: each one of the three organizations listed in the previous section may be a different organisation, but this is not always so. Sometimes two or more different profiles may coexist in the same person. However, it is very important that every person be fully aware of all his/her roles at every moment.

Conclusions.

In this paper we have presented the organisations that must take part in the software maintenance process, using as a framework the MANTEMA methodology.

Maintenance is the most expensive process of the software life cycle. One of the factors which influence the cost of any software process is the structure of the work team. An adequate distribution of the people can help to reduce the costs of such processes. Specially in maintenance, any costs reduction will be welcome. We think that the structure of organisations and profiles shown is useful for those enterprises concerned with software maintenance, as Atos ODS.

References.

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

[2] McClure, C. (1992) *The Three R's of Software Automation: Reengineering, Repository, Reusability*. Prentice-Hall, USA.

[3] McConnell, S. (1996) *Desarrollo y Gestión de Proyectos Informáticos*. McGrawHill-Microsoft Press.

[4] Emery and Emery (1975) *Participative Design-Work and Community. Center for Continuing Education, Australian National University.*

[5] Bayer and Highsmirth (1994). *RADical software development*, American Programmer, june, pp. 35-42.

[6] Brooks, F. P. (1995) *The Mythical Man-Month. Essays on Software Engineering Anniversary Edition*. Addison-Wesley, USA.

[7] 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*. Proceedings of the Third European Conference on Software Maintenance and Reengineering. Amsterdam (The Netherlands).

[8] ACM (1996). Communications of the ACM (contains a monograph about outsourcing), vol. 39, no 7

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

[10] Checkland, P. (1981). *Systems Thinking, System Practice*. Wiley, Chichester, U.K.

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

[12] Mazza, C., Fairclough, J., Melton, B., de Pablo, D., Scheffer, A. and Stevens,R. (1994). *Software Engineering Standards*. Prentice-Hall.

Book Reviews

The Netscape Programmer's Guide: Using OLE to Build Componentware Apps

Richard B. Lam

The Netscape Programmer's Guide is written by Richard B. Lam and published by Cambridge University Press, 1998, paperback, ISBN: 0-521-64820-3, 394 pp. $39.95

Dr. Lam has written a comprehensive tutorial on integrating a range of software products with Netscape's Navigator. The subtitle of this book is important, because a relatively small portion of *The Netscape Programmer's Guide* actually deals with the Netscape applications programming interface (APIA). The code samples and applications are all specific to the Microsoft operating environment, and depend especially on the use of DDE and OLE. The preface states explicitly that, "The book is aimed at software professionals programming for the Windows 95 environment." In fact, it is an excellent beginner's book for those with elementary programming experience in Visual Basic and C++. It should be applicable to any Win32 environment. Programmers who work primarily outside the Microsoft realm will find this book of marginal utility, but still informative. Those who are already experienced with implementing DDE and OLE servers may find about half of the book to be repetitive. The (by now obligatory) CD that comes with the book contains not only sources shown in the book, but also compiled versions of dynamic link libraries and executables, as well as associated project files. This is an excellent decision, since few programmers will have at their disposal the range of software used throughout the book.

After a brief introduction, the book proceeds immediately with construction of an OLE server in Microsoft's Visual Basic. The pattern in the book is to use Visual Basic as the primary language and C++ as the API implementation language, ostensibly due to reserved word conflicts with the Netscape API. (The code examples use Borland C++ 4.5). Dr. Lam keeps the coding straightforward and well explained. The C++ is self-contained, although extensive. Basic programmers could scan these sections and then just use the files on the CD as components. C++ programmers should have no difficulty in using a visual C++ tool to replace the Basic code. Once through the construction of an OLE automation controller, the next chapter takes up the use of the DDE interface. This is where the Netscape API is first formally introduced, but there is nothing special about the use of Netscape programming here. The next chapter returns to the use of the previous OLE work. It is here, on page 171 of 363 that the reader finally starts to do some specifically Netscape programming. A brief discussion of protocol handlers follows that. Chapter 7 describes how to connect to a Lotus Notes