Proceedings of the Third European Conference on

Software Maintenance and Reengineering

Chapel of St. Agnes, Amsterdam, The Netherlands
March 3-5, 1999

Sponsored by

Reengineering Forum Industry Associatioon

University of Amsterdam

University of Zurich

IEEE Information Technology Task Force (IT-IEEE)

IEEE Computer Society Technical Council on Software Engineering (TCSE)

IEEE COMPUTER SOCIETY

IEEE

Edited by
Paolo Nesi and
Chris Verhoef

Proceedings of the

Third European Conference on Software Maintenance and Reengineering

Chapel of St. Agnes, University of Amsterdam, The Netherlands
March 3 – 5, 1999

Edited by
Paolo Nesi and Chris Verhoef

IEEE COMPUTER SOCIETY

Los Alamitos, California

Washington • Brussels • Tokyo

IEEE Computer Society Order Number PR00090
ISBN 0-7695-0090-0
ISBN 0-7695-0091-9 (case)
ISBN 0-7695-0092-7 (microfiche)
Library of Congress Number 99-60126

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1-714-821-8380
Fax: + 1-714-821-4641
E-mail: cs.books@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1-908-981-1393
Fax: + 1-908-981-9667
mis.custserv@computer.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-kuTokyo 107-0062
JAPAN
Tel: + 81-3-3408-3118
Fax: + 81-3-3408-3553
tokyo.ofc@computer.org

Editorial production by Bob Werner

Cover art production by Joe Daigle/Studio Productions

Printed in the United States of America by The Printing House

Table of Contents

THIRD EUROPEAN CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING — CSMR 99

Session: Development and Maintenance

Session: Management

Session: Metrics

Session: Data Reengineering

Open Forum

Program Committee

Alain Abran, Universite du Quebec, Montreal, Canada
Vasu Alagar, Concordia University, Canada
Vincenzo Ambriola, Universita di Pisa, Italy
Ger Bakker, TriLoc Software Engineering Europe B.V., The Netherlands
Mark van den Brand, CWI, The Netherlands
Fernando Brito e Abreu, INESC, Portugal
Giacomo Bucci, University of Florence, Italy
Maurizio Campanai, CESVIT, Italy
Elliot Chikofsky, META Group, USA
Gerhard Chroust, Austria
Aniello Cimitile, University of Salerno, Italy
James Cross II, Auburn Univserity, USA
Luciano Da Fontoura Costa, IFSC — Universidade de Sao Paulo, Brasil
Juan Carlos Duenas, Universidad Politecnica de Madrid, Spain
Reiner Dumke, University of Magdeburg, Germany
Juergen Ebert, University of Koblenz, Germany
Alessandro Fantechi, University of Florence, Italy
Jean-Francois Girard, Fraunhofer Institute, Germany
Jean-Luc Hainaut, University of Namur, Belgium
John Harauz, Ontario Hydro, Canada
Brian Henderson-Sellers, Swinburne University of Technology, Australia
Even-Andre Karlsson, Q-Labs, Sweden
Taghi M. Khoshgoftaar, Florida Atlantic University, USA
Tereza Kirner, Federal University of Sao Carlos, Brasil
Phillip Laplante, Pennsylvania Institute of Technology, USA
Franz Lehner, University of Regensburg, Germany
Michele Marchesi, Universita di Cagliari, Italy
Thomas Marlowe, Seton Hall University, USA
Roland Mittermeir, Universita et Lagenfurt, Austria
Jean-Marc Morel, Bull S.A., France
Stefano Nocentini, IBM Semea, Italy
Mauro Pezze, Politecnico di Milano, Italy
Peter T. Poon, California Institute of Technology, USA
Lutz Richter, Universitaet Zuerich, Switzerland
Giacomo Sechi, IFCTR-CNR, Italy
Alex Sellink, University of Amsterdam, The Netherlands
Harry Sneed, SES, Germany
Ian Sommerville, Lancaster University, United Kingdom
Alexander D. Stoyen, 21st Century Systems, USA
Tooru Takeshita, Chubu University, Japan
Homa Taraji, Ernst & Young, LLP, USA
W.J. Toetenel, Delft Technical University, The Neterlands
Grace Tsai, Fairleigh Dickenson University, USA
Yoshinori Yamaguchi, Electrotechnical Laboratory, Japan

MANTEMA: a Complete Rigorous Methodology for Supporting Maintenance based on The ISO/IEC 12207 Standard[1]

Macario Polo, Mario Piattini, Francisco Ruiz, Coral Calero
{mpolo, mpiattin, fruiz, ccalero}@inf-cr.uclm.es
Grupo Alarcos
Escuela Superior de Informática
Universidad de Castilla-La Mancha
Ronda de Calatrava,5
13071-Ciudad Real (Spain)

Abstract

The maintenance of information systems is one of the greatest problems in the software life cycle: It is the most conflictive, costliest, less planeable; and the process requiring the most resources. In spite of this reality, most organizations do not possess methodologies for the maintenance.

The imperative need of controlling the maintenance process, have carried us to propose a maintenance methodology. In this paper an adaptation of the ISO/IEC 12207 standard for the processes of the life cycle maintenance is presented. This methodology is being used by ATOS-ODS, one of the most important European consultants on software outsourcing and maintenance.

Keywords: Life cycle, maintenance, software, software processes.

1. Introduction

Software maintenance is the most costly stage in the software life cycle. Some authors [4] affirm it absorbs between 67% and 90% of the total life cycle costs. On the other side, although most organizations have a methodology for software development, they do not follow any particular one for maintenance [12]. Consequently, a complete guide which helps maintainers to control software maintenance process is needed.

To define a software maintenance methodology, it is usêful to take into account more global frameworks, such as those defined for software life cycle processes [6] [7] [8]. These frameworks can also help organizations to define procedures and methodologies to achieve ISO 9000 certification.

This paper is organized as follows: in section 2 we justify the necessity of a well-defined Maintenance process. In section 3 we present MANTEMA, this new maintenance methodology. Our conclusions and future work outlines are discussed in section 4.

2. Justification of MANTEMA

Both ISO/IEC 12207 [8] and IEEE 1219 [5] present a set of activities and tasks that must be followed for software maintainers. However, none of them explicit which of them to do, depending on the maintenance type: all the activities are grouped and distributed in a sequential form. This lack of precision may difficult maintainers work, because there is no clarity about what activities must be done.

Although it is difficult to provide guidelines with universal validity, both standards expose the necessity of execute some type of activities and tasks selection, depending on the maintenance type:

- ISO/IEC 12207 propose the use of a Tailoring Process, that lets the adaptation of this standard to different software projects, taking in account variations on organizational politics and procedures, acquisition methods and strategies, system requirements, used life cycle, etc.
- IEEE 1219 advise "to classify the Modification Requests as corrective/adaptive/perfective/preventive and integrate them into sets that share the same design areas".

In the other side, [14] considers the necessity of establishing different flows of actions according to the maintenance type that must be applied. It is necessary to provide and validate methodologies which take in account specific characteristics of software maintenance organizations [3].

On the other side, it is well known [1] that software maintenance outsourcing is a fast growing activity, with an increasing number of organizations providing this

[1] This work is part of MÁNTICA project (CICYT 1FD-097-0168) and 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).

service. Certification is also more important in the current competitive market.

So, we find in this area a visible lack and necessity of clear methodologies for software maintenance.

We consider ISO/IEC 12207 [8] the most appropriate basis for defining a maintenance methodology because its growing importance in the software engineering community: As [11] remarks: "ISO/IEC 12207 will drive the world software trade and will impact maintenance".

ISO/IEC 12207 is a reference framework covering all aspects of the software life cycle processes. It describes the architecture of software life cycle processes, but does not detail how to implement the activities and tasks included in such processes. In ISO/IEC 12207, the activities which can be implemented during a software life cycle are separate in three groups: Primary processes (Acquisition, Supply, Development, Operation and Maintenance), Supporting processes (Documentation, Configuration management, Quality assurance, Verification, Validation, Joint review, Audit and Problem resolution) and Organizational processes (Management, Infrastructure, Training and Improvement). There is also a Tailoring process for adapting ISO/IEC 12207 to each specific case.

ISO considers Maintenance one of the primary processes of software life cycle. It is composed by six activities with some tasks in each one. A strict follow-up of this maintenance process may imply overlapping of some activities and tasks.

Furthermore, as we said previously, the existence of just one maintenance process for all possible types of maintenance may involve too time in the ISO/IEC 12207 tailoring process. So, we think it would be adequate to modify ISO/IEC 12207 to adapt it specifically for maintenance. In the next section we present a methodology incorporating these ideas.

3. MANTEMA: a methodology for software maintenance.

In this section we present a new maintenance methodology, "MANTEMA", built from ISO/IEC 12207. In this methodology, which is a complete one, Maintenance consists of a list of activities and tasks and two sets of processes taken from ISO/IEC 12207: the first of these sets is composed by some processes integrated in the own maintenance process (acquisition, supply, development, documentation, verification, validation, joint review, audit, problem resolution and management); the other set is composed of auxiliary processes, which are used by the maintenance activities and tasks when they are needed.

Furthermore, MANTEMA provides a set of document templates to help to the managing and controlling of all the maintenance activities and tasks.

We also define who are the parts that are involved in the maintenance process, as it is shown in section 3.3.

We want that this methodology convert the maintenance in a full-controlled process and with continuous improvement. So a set of metrics is proposed in certain activities and tasks.

3.1 General view of MANTEMA

MANTEMA consists of specific activities and tasks for each maintenance type, except two sets of initial and final tasks, common for all types. In Figure 1 we can see a graph to illustrate these ideas. Five types of maintenance are defined:

1) *Urgent corrective*: a detected error prevents normal system operation and the solution time is critical.

2) *Non-urgent corrective*: a detected error does not block the normal operation of the system and the solution time is not critical.

3) *Perfective*: when new functionalities are added to the system.

4) *Preventive*: consists of the software modification to improve its maintainability and quality properties.

5) *Adaptive*: when the system will change its execution environment.

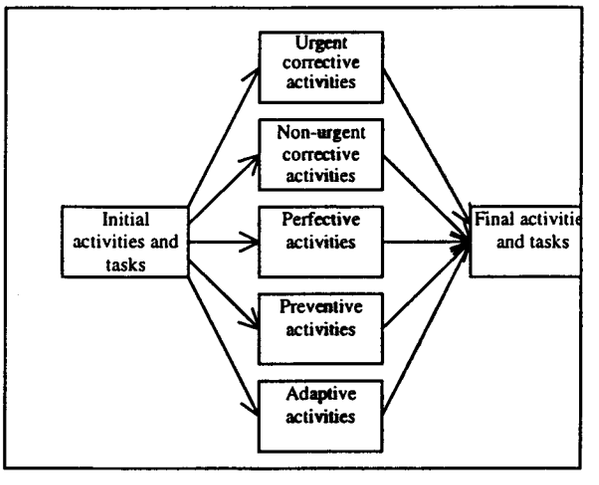

Figure 1. General view of MANTEMA.

3.2 One example: Urgent corrective activities and tasks

Next table details urgent corrective activities and tasks. This is the most urgent maintenance type. So, it is the one with the minor number of "bureaucratic" tasks. Among others, it has a special difference with the other maintenance types: the first task that is made when the "Corrective intervention" is finished is dedicated to put the corrected software product into the production environment. Following, documentation is completed and the intervention is closed.

The structure of the other maintenance types is similar, but its contents differ substantially: all of them

	URGENT CORRECTIVE ACTIVITIES AND TASKS						
	Error analysis	Corrective intervention			Intervention close		
	1.1 Investigating and analysing causes	2.1 Making corrective actions	2.2 Complimenting documentation	2.3 Verification of the modifications	3.1 Putting the software product to production environment	3.2 Documenting correction	3.3 Saving intervention
Inputs	Software product in operation Urgent error Modification request	Software to be corrected	Old software (with errors) New software (without visible errors)	Corrected software Unitary test cases Integration test cases	Corrected software	Documentation of: · Corrective actions · Tests · Customer conformity	Full document
Outputs	Software to be corrected	Corrected software	Documentation with the corrective actions made	Assurance of the correction of the error	Corrected software in operation	Full document	Stored full document
Responsible	Maintenance team	Maintenance team	Maintenance team	Maintenance team	Maintenance team	Maintenance team	Maintenance team Maintenance responsible
Interfaces with other processes					Configuration Management		

are structured with three activities: first, an analysis of the modification request is done; afterwards, the intervention and tests are executed; the third and last activity is the close of the intervention.

3.3 Participant organizations

In the software maintenance process, three organizations are consider:

1) Customer, organization which owns the software and requires the maintenance service.

2) Maintainer, the organization which supplies the maintenance service. The maintenance team is composed by the real persons who implement the maintenance tasks.

3) User, the organization that uses the maintained software.

Each one of this organizations may be a different organization, but this is not so always.

Based in [5] and in [10], we define a set of profiles for every participant organization:

· Customer profiles.

- Petitioner: who promotes a Modification Request and establishes the need requirements for its implementation and tell them to the maintainer.
- System organization: is the department that knows well the system that will be maintained.
- Help-desk: is the department which attends to the users. It also reports to the petitioner with the incidences remitted by the users to generate modification request.

· Maintainer profiles.

- Maintenance-request manager: decides whether the modification requests are accepted or rejected and what type of maintenance should be applied.
- Scheduler: must plan the accepted modification-request queue.
- Maintenance team: is the group of persons who implement the accepted modification request.
- Maintenance responsible: prepares the maintenance stage. Also establishes the standards and procedures to follow with the maintenance methodology used.

· User profile.

- User: uses the maintained software. He communicates the incidences to Help-desk.

3.4 The Common initial activities and tasks

There are three activities in this group: the initial study, the process implementation and the study of the modification request. All of them are important but, for those organizations which maintain third-party software, the first one is specially relevant.

When there is outsourcing, the maintenance organization must contract it with the customer in this activity. Before the acceptation of a maintenance project, we must to assess its risk probability. Working with Euromethod and with the ATOS-ODS own experience, we have constructed the "Risk questionnaire" to evaluate risks and to know the degree of external dependence of the Customer.

The first task of the second activity ("Process implementation") is the "Knowledge acquisition"; here, the Maintainer learns all sort of details of the software product: studying the documentation, watching how the Customer maintenance team is working, etc. In this case, when the Maintainer is an external organization, the Customer feels that the Maintainer is not really working: he is only seeing an intrude. To avoid this impression, when this activity is finished, the Maintainer must give to the Customer a good documentation with analysis, audit reports, possible improvements, etc. There are more tasks in this activity: Developing plans, Defining request modification procedures, Implementing configuration management process and Preparing tests environments.

The third activity (Study of the modification request) is composed by two tasks: "Reception of the modification request" and the "Decision about the type of intervention". In the last one, the Maintenance-Request manager must decide the type of maintenance to be applied. The experience demonstrates that the means by which the Modification Request arrive determines, in many cases, the maintenance type that will be applied: for example, when the request is made by telephone, Maintainer knows quickly that is an urgent corrective maintenance. Fur-

thermore, the Maintenance-request manager has sometimes a very limited decision power, specially when the Customer is an external organization.

3.5 Maintenance documents

During the maintenance process, a great quantity of documentation is generated. In MANTEMA, we provide standard templates for each generated document as, for example:

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

3.5 Metrics

To achieve the continuos improvement of maintenance process, metrics must be used in almost all the activities and tasks of the process.

Product metrics are important to observe the evolution of maintained software. In general, it is good that the values of these metrics (cyclomatic complexity [9], number of errors, etc.) decrease. Furthermore, historic data of product metrics are used to estimate maintenance costs using mathematical analysis (linear regression, for example).

As most of product metrics are designed for programs, but as the importance of the databases on the information system complexity is increasing, we have also propose a set of metrics to assess the complexity of relational databases [13].

There is also a set of metrics to be used specifically in maintenance process: service indicators (Response Time, Respect to Planification), Flexibility with no replanification (number of persons-month we can dispose with no replanification nor new resources allocation), Number of hours replanificated, Complexity evolution, Ratio of requests and efforts by each way (urgent, non-urgent, etc.), etc.

For concluding this section, we cannot forget the particular case of preventive maintenance, when big maintenance works are made and when reengineering and reverse engineering activities are needed [14]. Sneed [15] proposes an original and useful method that can be applied to classify our application portfolio when we are thinking about the beginning of a reengineering project.

4. CONCLUSIONS AND FUTURE WORK

In this work we have presented MANTEMA, a methodology for fully supporting software maintenance. As we have justified, there is a lack in this software engineering area.

MANTEMA incorporates a well-defined set of tasks according to the maintenance type, supporting documentation control.

At the moment, we are working in a evolved version of MANUTEN-CONDUCT, an automatic tool for integral support of maintenance developed by ATOS-ODS. an automatic tool for integral support of maintenance developed by ATOS-ODS. This tools must saves data relative to products and processes. In a very next future we will propose metrics specifically for maintenance process.

5. REFERENCES

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

[2] Albretch, A.J. *Measuring application development productivity*. Proc. of the IBM appl. Develop. Symp. Monterrey.

[3] Basili, V., Briand, L., Condon, S., Kim, Y., Melo, W. y Valett, J.D. *Understanding and predicting the process software maintenance releases*. Proceedings of the International Conference on Software Engineering. IEEE, 1996.

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

[5] IEEE Std 1219-1992. Standard for Software Maintenance.

[6] IEEE Std 1074-1995. Standard for Developing Software Life Cycle Processes (ANSI).

[7] IEEE Std 1074.1-1995. Guide for Developing Software Life Cycle Processes (ANSI).

[8] ISO/IEC 12207. Information Technology. Software life cycle processes. 1995.

[9] McCabe, T. *A software complexity measure*. IEEE Transactions on software engineering, vol. 2, no. 4, pp. 308-320.

[10] Mazza, C., Fairclough, J., Melton, B., de Pablo, D., Scheffer, A. Y Stevens,R., *Software Engineering Standards*. Ed. Prentice-Hall, 1994.

[11] Pigoski, T.M. *Practical Software Maintenance*. Wiley Computer Publishing. New York, USA, 1996.

[12] Piattini, M.G., Ruiz, F., Polo, M., Villalba, J., Batanchury, T. and Martínez, M.A. *Mantenimiento del software: conceptos, métodos, herramientas y outsourcing*. RAMA. Madrid (Spain), 1998.

[13] Polo, M., Piattini, M.G., Ruiz, F. and Calero, C. *Métricas de calidad y complejidad para bases de datos*. Actas de las III Jornadas de Ingeniería del Software. Murcia (Spain), Nov. 1998.

[14] Pressman, R.S. Software engineering. *A practioner's approach*. McGraw-Hill, 1993.

[15] Sneed, Harry M. *Planning the Reengineering of Legacy Systems*. IEEE Software, vol. 12, nº 1, Jan. 1995.