Evaluating CM^3: Problem Management

Mira Kajko-Mattsson

Dept. of Computer and Systems Sciences, IT University, Forum 100,
SE-16440, Kista, Sweden
mira@dsv.su.se

Abstract. *CM^3: Problem Management* is a detailed problem management process model to be utilised within corrective maintenance. It has been developed at ABB and evaluated for its industrial relevance within 17 non-ABB organisations. In this paper, we present the evaluation results of *CM^3: Problem Management.* Our primary goal is to confirm its industrial relevance. Our secondary goal is to establish current state of problem management practice in the industry in order to provide a basic reference point from which the desperately needed research into software maintenance can proceed.

1 Introduction

Software problem management process is one of the most important processes within corrective maintenance. It not only manages software problems, but also provides a basis for quantitative feedback important for assessing product quality and reliability, crucial for continuous process analysis and improvement, essential for defect prevention and for making different kinds of decisions. Mature problem management process is a prerequisite for achieving CMM Level 5 [1].

Corrective Maintenance Maturity Model (CM^3): Problem Management is a detailed problem management process model. It has been developed at ABB and evaluated for its industrial relevance within 17 non-ABB organisations. In this paper, we have chosen some of the process requirements inherent in *CM^3: Problem Management* and matched them against the state of practice in software organisations. Our primary goal is to confirm the industrial relevance of *CM^3: Problem Management*, that is, to find out whether our model is realistic, down-to-earth, and whether it properly reflects current industrial practice. Our secondary goal is to establish current state of problem management practice in the industry to provide a basic reference point from which the desperately needed future research into software maintenance can proceed.

2 Methodology

CM^3: Problem Management was developed at two ABB organisations: *ABB Automation Products* (ABB APR) and *ABB Robotics (ABB ROP)*. Due to the scarcity

A. Banks Pidduck et al. (Eds.): CAISE 2002, LNCS 2348, pp. 436-451, 2002.

of relevant literature about this domain, it is entirely based on our empirical study of industrial processes.

Table 1. Organisations contributing to the evaluation of *CM³: Problem Management*

Name	No of employees	Products/Services	Software size
Ericsson Radio Systems	14 000	Base stations	Impossible to assess
Ericsson Telecom AB	300 (75 maintainers)	Telecommunication systems	ca 100 000 LOC
Ericsson Global IT Services	900 (5 maintainers)	Telecommunication products	ca 120 000 LOC
Ericsson Infotech	600	Sw processes and tools	Impossible to access
Ericsson AB	26 000	platform for AXE switches	400 blocks
ABB Automation Products	1400	Process control systems	Impossible to assess
ABB Robotics	600	Industrial robot systems	3 000 000 LOC
European Space Agency	3-4 maintainers/interviewed departm.	Space robot arm	250 000 LOC
SAAB AEROSPACE	6000 (400 maintainers)	Software for military aircraft	Impossible to assess
Nexus	300	Embedded systems	10000-1000000 LOC.
Postgirot Bank AB	3200 (170 maintainers)	Financial systems	6000 programs in 168 different systems
eumetrix Financial Solutions	55	Financial Systems	500 000 LOC
Navigera Business Consulting	47	Business systems	No size provided
Foyer Consulting AB	1	Various types of systems	Strongly varying
On, Line	20	Administration systems	700 000 LOC
Sema Group Infodata	300 (17 maintainers)	Information systems	142 programs
VM Data Public Partner	70 maintainers/depart.	Financial systems	Impossible to assess
Försvarets Radio Anstalt	ca 1000	Support systems	Impossible to assess
Tele2	6000	Support systems	250 000 LOC /interv. dep.

CM³: Problem Management was evaluated both within and outside ABB. Seventeen non-ABB organisations agreed to expose their process models to our study. When choosing them, we have attempted to cover the wide spectrum of the today's IT state. Deliberately, we have chosen different sizes and different types of software organisations. The types of products maintained by these organisations varies from financial systems, business systems, embedded real-time systems, consulting services, administrative systems, and different types of support systems. The evaluation organisations are presented in Table 1.

The evaluation was conducted against a specific evaluation model presented in Appendix A. This model consists of the interview questions reflecting all the process steps inherent in *CM³: Problem Management*. For learning about our process model, we advice our reader to study [2].

The fine-grained nature of our questions makes our interview results highly sensitive. Our results reveal the detailed state of practice of the organisations studied. To ensure the anonymity of the organisations studied, we have grouped the results with no references to a particular organisation.

It is not easy to evaluate a process model within the industry. Many industrial processes are still too coarse-grained. They have not implemented all the process steps suggested by our model. When evaluating *CM³: Problem Management*, we first checked whether a certain process step was fully implemented by the industrial processes. If not, then we checked whether this step was logically followed. This sufficed to fully defend its implementation.

3 Evaluation Results

Seventeen non-ABB organisations exposed their processes to our evaluation. One of these organisations was a consultant company evaluating our problem management process within about ten organisations. Since no exact quantitative feedback could be provided for these ten organisations, we have excluded them from our quantitative analysis. These organisations are however included in our qualitative analysis.

Our two ABB organisations are also involved in our evaluation. Presently, ABB APR is in the process of changing its problem management process, but the old process is in use as well. For this reason, we treat the two ABB APR processes as two separate ones. When presenting their results, we refer to them as to two separate organisations. Summing up, nineteen processes are involved in the quantitative evaluation of *CM³: Problem Management* (three ABB and 16 non-ABB processes; one non-ABB organisation was excluded from our quantitative study) and thirty processes are involved in the qualitative one.

Due to the fact that our model is very comprehensive, we cannot present the evaluation results for all process steps. We have chosen only a subset of them. We have also excluded all kinds of motivations for choosing *CM³* process steps. Instead, we advise our reader to study [2]. Each process step presented below has a numerical reference to the interview question(s) listed in Appendix A. Our evaluation results are presented in Sections 3.1-3.11.

3.1 Process Definition and Process Visibility

We have checked whether the organisations studied have defined and documented their problem management processes, whether they have divided them into process phases, whether they have recorded the results of these process phases, and finally, whether they record additional activities that have not been predetermined by their defined problem management processes.

Problem Management Process Definition (see I:Q.1): Sixteen out of 19 organisations have defined and documented their problem management process. The interviewees from two of the three remaining organisations claim that they have defined a problem management process but they have not documented it as yet. Finally, the third organisation has no process model defined at all. Maintainers in this organisation use their own personal methodology based on the experience gained.

Process Visibility (see I:Q.2 and I:Q.3): Sixteen out of 19 organisations divide their problem management processes into several phases. This division however, is coarse-

grained. Usually, it consists of the following phases: *Problem Reporting, Problem Analysis,* and *Problem Resolution.* Only four of these 16 organisations further divide these phases into sub-phases and activities. These four organisations have also defined process models for the major problem management activities such as *Problem Investigation, Problem Cause Identification, Modification Design* and *Modification Implementation*. These process models are usually represented in the form of process guidelines. Six other (out of 16) organisations have expressed a strong wish for process models for each major problem management activity. Two of these six are presently working on fine-graining their problem management processes and on defining process models for major activities.

Recording the Results of Process Phases (see I: Q.4.1 and Q.4.3): Fifteen out of 19 organisations record process data concerning the results of the process phases. This data is however very coarse-grained. Usually, it encompasses the combined results of the following group of major activities: *Problem Investigation, Problem Cause Identification, Modification Design*, and *Modification Implementation.* The information being recorded varies. Usually, it covers effort, resources, modification size, impact, and experience gained. Only two of the 15 organisations record the results of each individual process activity.

Recording Additional Activities Not Predefined by the Problem Management Process (see I: Q.4.2): Four out of 19 organisations record the additional activities that have not been predetermined by their problem management process. Only major additional activities are recorded such as a visit to a customer site.

3.2 Process Analysis and Control

In this section, we report on the types of activities that are conducted in order to analyse and control the problem management process. We have checked the following: (1) measurement of the problem management progress, (2) rules for recording problems in problem reports, (3) management of duplicate problem reports, (4) control of the correctness of the problem report data, (5) distinguishing between internal and external problem reports, (6) categorisation of maintenance requests, and finally, (7) comparison of plans to the actual results.

Progress of Problem Management (see II: Q.1): Sixteen out of 19 organisations measure the progress of problem management process using status values and the dates when these status values change. This information is primarily used for monitoring the problem resolution process and workload of individual engineers. It is also used for controlling the amount of corrective work that remains to be done for a certain release.

Rules for Recording Problems in Problem Reports (see II: Q.2 and Q.3): Sixteen out of 19 organisations studied identify all maintenance requests (problem reports in our case) by assigning a unique identifier to each reported problem. Thirteen out of 19 organisations describe one problem in one and only one problem report. If a problem report communicates several problems or if an analysis of a problem reveals several underlying problems, then all these problems get reported anew in separate problem reports. The six remaining organisations sometimes report on several problems in one

problem report. Usually, these problems are somewhat related. The interviewees coming from these organisations admit that this practice is clumsy, severely obstructing problem management.

Management of Duplicate Problem Reports (see II: Q.4, Q.5, and Q.6): Sixteen out of 19 organisations distinguish between unique and duplicate problems. If several problem reports relate to the same problem, then usually one report becomes a master report containing all information about the unique characteristics of the problem. This master report is continuously updated with relevant information from the duplicate problem reports. The duplicate problem reports get closed, but remain related to the master report.

Two of the organisations simultaneously attend to all the duplicate problem reports in a group. They do so with the motivation that these reports come from different customers and hence may contain information that differs from customer to customer, that they wish to keep in separate problem reports.

All the sixteen organisations continuously revise the uniqueness of the problems throughout the process. One of the interviewees has mentioned that sometimes as late as during the modification implementation, the maintainers may discover that the problem being attended to is a duplicate problem.

Control of the Correctness of the Problem Report Data (see II: Q.2 and Q.3): Seventeen out of 19 organisations continuously check the correctness of the problem report data throughout the whole problem management life cycle. Eighteen out of 19 organisations check the quality of the reported data before assigning the problem report to the maintenance team. This is usually conducted by a role corresponding to *CM³ Problem Report Administrator, CM³ Problem Report Owner*, or a *CM³: CCB*.

Distinguishing between Internal and External Problems (see II: Q.9 in Appendix A): Ten out of 19 organisations studied actively classify problem reports as internal or external. This information is utilised for different purposes such as statistics, planning, evaluation of customer satisfaction, and measurement of effort. Five of the remaining nine organisations may record this information, but they do not use it for any particular purpose. For one of the remaining four organisations, this information is not relevant. They only maintain internal support systems.

Identifying Maintenance Category (see II: Q.10, Q.11, and Q.12): All the organisations studied identify a maintenance category for each maintenance request. In our context, this means that all software problems are classified as corrective maintenance, to be distinguished from perfective and adaptive categories. Seven of 19 organisations studied conduct some form of preventive maintenance. By this, they mainly mean restructuring some parts of the software system. Sixteen of the 19 organisations studied continuously revise the maintenance category during the problem management life cycle.

Comparing Plans to Actual Results (see II: Q.18): Eleven out of 19 organisations compare the plans to the actual results. They do it for learning lessons and for providing feedback for process improvement. The maintainers within the remaining seven organisations do not make any comparisons at all.

3.3 Managing Software Products

In this section, we report on the mechanism used for identifying the software products affected by the reported problems, utilisation of the information on the product environment at both the customers and maintainers' sites, recording practice of the problem occurrence date, and finally, the designation of releases in which problems will be investigated and in which problem solutions will be implemented.

Identifying Software Products (see III: Q.1, Q.1.1, Q.1.2, and Q.1.3): All the organisations studied identify the products in which software problems are encountered. They usually report the *Release Id* or *Article ID*, a name which is then translated into the *Release Id* by the maintenance organisation.

The identification of the software systems is not always so straightforward when reporting on external software problems. Some systems are very complex. They may be real-time embedded systems, integrated with other systems, and these systems may not always be produced by one and the same organisation. In such a case, it is not always easy to identify the system and its release at the problem encounter. The organisations possessing such systems have chosen a special procedure for system identification. Usually, representatives from different systems have a meeting or a series of meetings during which they attempt to analyse the problem and localise it within such a complex system.

Only in twelve organisations out of 19, external submitters are able to identify the software product on a more detailed level than its *Release ID*. Usually, they may identify some major functionality or sub-component. In the remaining processes, more detailed identification of the product cannot be made by the external submitters.

Concerning the reporting of software problems internally, problems may be reported on as detailed level as module- and/or even module line level in most of the organisations studied. This, however, depends on the problem nature and on the individual problem submitter.

Identifying Product Environment (see III: Q.2, Q.2.1, and Q.2.2): Eighteen out of 19 organisations studied have access to the data on the environment of their customers. They either have this information available at the maintenance execution level or they are provided with it by the upfront maintenance level (support). We have not checked, however, whether this information is easily available at upfront maintenance. Due to the specific nature of their product, the remaining 19th organisation contacts its customers for finding out this information, when necessary.

Date and Time when the Problem was Encountered (see III: Q3 and Q3.1): Ten out of 19 organisations record the date and time when the problem was encountered. This value may also be easily recreated from log files. Only two of the organisations studied use this value in the reliability measurements. Primarily, the organisations studied use this value to learn about the problem, for instance, to find out whether it was time related. Some organisations use this value to study the log files before the problem was encountered in search of problem symptoms.

Designating Software Releases in which the Problem is Investigated (see III: Q.5 and Q.5.1): Thirteen out of 19 organisations carefully choose the versions in which the software problem is to be investigated. First, they usually recreate the problem in

the version of the product in which the customer encountered the problem. Second, they choose the release(s) in which it is relevant to identify the reported software problem. This choice is usually determined by many varying factors such as the number of releases being presently utilised by the customers, customer status and business criticality, the severity of the problem, and so on.

In the remaining six out of the 19 organisations, the designation of the appropriate version is not an issue. These organisations usually attend to software problems in the latest releases. Therefore, they investigate the software problem first in the release in which the software problem was encountered (irrespective of its age) and the latest release in order to find out whether the problem still exists. It may happen that the problem has disappeared due to changes in the earlier releases.

Designating Software Releases in which the Problem Solution will be Implemented (see III:Q.4, Q.4.1, and Q.4.2): Thirteen out of 19 organisations designate versions of the software system in which the problem will be implemented. Usually, more than one release may be designated for this purpose. These organisations also continuously revise the designation of these software releases. The continuous revising is necessary for a meaningful planning and prioritisation of problems as new software problems (with higher severity and priority) are reported or as maintenance engineers gain better understanding of the problem.

Five of these thirteen organisations make patches. They do so mainly in cases when it is urgent to solve a problem. These patches, however, are soon transformed to new releases. In the remaining six out of 19 organisations, the choice of the version in which the problem will be implemented is not relevant. This is because they either have only one customer or they have a policy that they never do changes in the older releases. If their customers have encountered a problem in some older release, then they must simply upgrade their systems if they want the problem to be solved.

3.4 Recording Problem Reports (see IV: Q.4 and Q.4.1)

Sixteen out of 19 organisations studied record all their problems in an organisation-wide problem report repository and tracking systems. The remaining three organisations do it partially. Some problems internally encountered may never get reported in these three organisations. They may be communicated orally via the telephone or email. The first of these three organisations does not record minor problems of the cosmetic type. In the second organisation, recording discipline depends on the individual maintenance engineer. In the third organisation, the problems that are usually reported via the mailing tool do not get stored in the organisation-wide repository. In this organisation, all problem reports disappear as soon as they get resolved. Hence, no historical data may be extracted. Finally, the consultant organisation not included in our quantitative analysis claims that there are still organisations that record their problem reports in a paper form.

3.5 Describing Software Problems

We have checked whether the organisations studied provide guidance (templates) to their problem submitters on how to describe software problems when reporting them,

whether the organisations submit reports on problems encountered during maintenance work, and, whether the organisations continuously revise the descriptions of software problems during the problem management process.

Templates for Describing Software Problems (see V: Q.1, Q.1.1-1.8, and Q.2): Sixteen out of the 19 organisations studied provide a template for problem descriptions. This template, however, differs in the number and choice of predefined constituent data fields. The following data fields are provided:

General problem description: All the organisations studied allow their submitters to provide a general description of a software problem.

Problem effects and consequences: Nine out of 19 organisations studied provide a predetermined data field in a problem report for describing problem effects and consequences. In the remaining ten organisations, this information may be communicated as part of the general problem description.

Problem symptoms: Only three out of 19 organisations actively collect information on the problem symptoms using a predetermined data field in a problem report. Four other organisations treat symptoms as consequences. The remaining twelve organisations may communicate them as part of the general problem description. The information on symptoms is utilised for problem investigation. Symptom information provides important feedback when finding out what exactly happened before the problem occurred. Symptom information is also used as a checklist for the identification of duplicate problems. Identification of duplicate reports using the symptom information is conducted manually by the organisations studied.

Problem conditions: Five out of 19 organisations provide predetermined fields for describing problem conditions. One of those five organisations provides a combination of several predetermined fields utilised for the purpose of describing problem conditions. The remaining 14 organisations may or may not collect this information as part the general problem description on a case to case basis. By problem conditions, the organisations studied mean the specific date and time of the problem occurrence (e.g., a year shift, or peak time), a combination of applications active during the problem occurrence, cargo, weather conditions, exact position of the product (if embedded product), and other things.

Problem Reproduction: Ten out of 19 organisations provide a predefined field for a description of how to reproduce the reported software problem. This field is also utilised for describing alternative execution path(s) to the problem. Out of these ten organisations, five classify problems as reproducible and irreproducible. The remaining nine organisations may provide this information as part of the general problem description. One out of these remaining nine organisations provides a predetermined field for classifying whether the problem is reproducible or not.

Attachments: Sixteen out of 19 organisations manage attachments such as log files, application programs, error messages, screen dumps, data base contents. Only one of these 16 organisations manages attachments manually, whereas the remaining 15 do it electronically.

Submitting Additional Problem Reports (see V:Q.6): Twelve out of 19 organisations always report on additional problems encountered during the problem management process. In these organisations, the engineers are not allowed to attend to problems without first reporting on them, even if these problems imply trivial

changes. Three out of the seven remaining organisations never report on software problems encountered internally. They attend to these problems without reporting them. At most, this information may be visible in the release documentation. One of these organisations admits that it is a bad habit. They feel that they must change it, because they cannot get the whole picture of the changes in their systems. Also, if a new problem is encountered, then they cannot see whether it has already been resolved. In the remaining four out of seven organisations, reporting on additional software problems varies as follows:

If the encountered problem is not related to the problem that is currently being managed, then a new problem report is created. Otherwise, it is described in the problem report being currently attended to. The organisation admits that this practice creates difficulties especially in cases when the problem substantially expands. It then becomes difficult to manage and control.

If a similar problem is discovered during attendance to some problem, and it implies very little change, then it is not reported but corrected under the umbrella of the attended problem. All other similar problems must be reported in separate problem reports.

Revising Problem Report Data (see V:Q.5): Seventeen out of 19 organisations continuously revise the problem report data as they gain more understanding about the problem. This step however is usually not an explicit part of the maintenance methodology of the organisations studied. Concerning the two remaining organisations, this step is up to each individual maintainer. The interviewees from these organisations agree that an explicit statement of this step is essential for increasing productivity and making problem validation and verification effective.

3.6 Problem Investigation

We have checked whether the main activities within the *Problem Investigation* phase have been implemented. They are (1) study of a problem report, (2) study of a software system, and (3) recreation of the problem.

Study of Problem Report (see VI: Q.1, Q.1.1): All the organisations studied include studying of a problem report as the first step within the *Problem Investigation* phase. The duration of this step varies from half an hour to several months. The longer time usually arises when the maintainer must complement the problem report with additional data by contacting the problem submitter. Please observe that many of the organisations studied contact problem submitters via the upfront maintenance (support) process.

Study of Software System (see VI: Q.2, Q.2.1, Q.2.2): All interviewees claim that their maintenance engineers study the software system when investigating the problem. Twelve of the interviewees claim that their maintainers fully understand the part of the software system they are maintaining. However, most of them have no process feedback for supporting this statement. The remaining seven interviewees admit that their maintainers have problems with understanding the software systems. In one of them, the second root cause to the externally reported software problems is lack of maintainers' knowledge of a software product.

Lack of knowledge may arise in cases when the system has been developed by a third party (another organisation), or the system is so complex that even a limited set of components can be difficult to understand. In one organisation, the maintainers only learn the parts of the product required for understanding the problem and for infusing changes without acquiring the knowledge of its remaining parts.

To study a software system may take time. The time span required for conducting this process step varies from half a day to several months. In cases when it takes a longer time, the maintainer is a novice, or she must study several releases of the same product and identify their differences, or the nature of a problem is such that it requires a meticulous study of a software system, for instance, in embedded systems when one must check all the interfaces to hardware, or there is inadequate documentation so that reverse engineering needs to be performed.

Recreating Problems (see VI: Q.3, Q.3.1): When recreating software problems, the organisations may follow problem descriptions, simulate software, or define test cases and test software. Seventeen out of 19 organisations studied define test cases when investigating software problems. All except two of those 17 organisations document these test cases. The remaining two organisations (out of 19) do not create any test cases. They just follow the problem descriptions when recreating problems.

3.7 Problem Cause Identification

We have checked whether the main activities within the *Problem Cause Identification* phase have been implemented. They are (1) identification and recording of defects, and (2) classification of defects.

Identification and Recording of Problem Causes (see VII: Q.2, Q.2.1, Q.2.2, Q.2.3): When identifying causes, all the organisations studied prescribe a study of the pertinent documentation items and a control of their correctness. Sixteen of them record all the documentation items in which defects were found. Usually, these documentation items are identified on a module granularity level. Only three of those 16 organisations record defects on the software code line level. One of the three remaining organisations (out of 19) only records this information for problems whose defects are difficult to identify. The remaining two organisations do not implement this process step at all.

Classification of Defects (Problem Causes) (see VII: Q.3, Q.4, Q.5): Eleven of the 19 organisations studied classify the defects (problem causes). This information is used for planning future work and for different types of statistical analyses. Five out of these eleven organisations use it for identifying deficiencies (root causes) in the organisational processes.

3.8 Root Cause Analysis (see VIII: Q1, Q.2, Q.3, Q.4)

We have checked whether the organisations studied attempt to identify deficiencies in the process leading to the defects. Only five out of 19 organisations studied conduct root cause analysis. However, they do not do it on every software problem. The

candidates for root cause analysis are either external software problems, very serious internal and/or external ones, or groups of very frequently encountered problems.

3.9 Modification Design

We have checked whether the organisations studied create mental pictures of problem solutions in the early phases of the problem management process, whether the organisations create suggestions for problem solutions before implementing them, whether they evaluate and inspect/review these suggestions, and whether they plan for implementing them.

Mental Pictures of Problem Solutions (result of questions see IX: Q.1, Q.2, Q.3): Design of problem solutions should start as soon as the problem gets reported to the maintenance organisation irrespective of how much detail is provided about it. All organisations except for one (18 organisations), create a mental picture of what a problem solution might look like. This mental picture provides an important feedback for making important decisions through the problem management process. It is also continuously revised as more understanding is gained about the problem solution. Only two of those 18 organisations document the mental pictures of problem solutions.

Suggestions for Problem Solutions (see IX: Q.1, Q.2, Q.3): Fifteen out of 19 organisations may create one or several modification designs (problem solutions) for resolving a problem. Usually, these organisations create one problem solution. Several alternative problem solutions are only created for major and/or more severe problems, especially for those implying expensive changes. Four of these 15 organisations always attempt to create several problem solutions, if possible. But, only one suggestion is recorded in the problem report repository and tracking system.

The remaining four organisations only create one problem solution. One of them has a time constraint to deliver a problem solution as quickly as possible. From the time when the problem cause (defect) has been identified, the clock starts ticking and the maintainers have only 48 hours to resolve the problem. Hence, there is no time for creating alternative solutions.

Evaluating Problem Solutions (see IX: Q.4, Q.5, Q.6): Within 18 out of the 19 organisations, maintainers evaluate their own suggestions for problem solutions. When evaluating them, they consider these factors: modification size, change impact, ripple effect, effect on customers, effort and resources required for solving the problem, the feasibility of solving the problem, and the benefits and drawbacks of solving the problem.

Inspecting and/or Reviewing Suggestions for Problem Solutions (see IX: Q.7, Q.8): Before being presented to the CCB, the suggestions for problem solutions should be inspected and/or informally reviewed. Within 15 out of 19 organisations, inspections and/or informal reviews are being conducted. The four remaining organisations neither inspect nor review their suggestions for problem solutions.

The level of formality varies in the 15 organisations depending on the size, severity and complexity of the software problem. Usually, major problems are formally inspected and minor problems are only informally reviewed. It may also happen that

suggestions for solutions of minor problems are not reviewed at all. In four out of the 15 organisations, all suggestions, even those for trivial changes, are always inspected and/or informally reviewed.

Planning for Implementing Problem Solutions (see IX: Q.9, Q.10): Eighteen out of 19 organisations make plans for implementing problem solutions, but only 16 out of those 18 record these plans. The remaining 19th organisation does not make any plans at all (time is ticking and the organisation must resolve the problem as soon as possible).

For one of the 18 organisations, planning means designating the date when the problem solution should be ready. In the remaining 17 organisations, the plans encompass time schedule, designation of the versions of the software system in which problem solutions will be implemented, effort and resources required such as person-hours and equipment to be available during a certain period of time, prerequisites for implementing the solution such as closure of the system for a certain period of time, and a time schedule.

3.10 Modification Decision

We have checked whether the organisations studied have institutionalised a practice for making decisions on all corrective changes in the software product. Two types of decision making have been identified: (1) management decision where the maintenance engineers' closest managers are involved in following the problem management process and in making intermediate decisions important for its further progress, and (2) CCB decisions where *Change Control Board* members make the most important decisions on changes to software systems.

Management Decision (see X: Q.1, Q.2, Q.6): Within 15 out of 19 organisations, the maintainers report on the results of major problem management phases to some authority for control and for decision making on the future progress of the problem. Usually, correspondences to the CM3 role *"Problem Report Engineer"* report to the correspondences to the CM3 role *"Problem Report Owner"*. This reporting is however scarcely documented. It is mainly managed orally. At the most, its results may be visible in the change of the problem report status values, their dates, and some comments. Within four remaining organisations (out of 19), this reporting does not take place at all. Maintainers are allowed to make their own decisions during the problem management process.

CCB Decision (see X: Q3-Q.6): Within 16 out of 19 organisations, the most important decisions are made by a *Change Control Board.* These decisions concern the progress of serious problems and changes to software. In 14 organisations (out of 16) all changes must be permitted by the CCB. In two organisations (out of 16), only major problems are paid heed to. CCB chooses a suggestion for a problem solution on the basis of the evaluation results conducted by maintainers themselves, inspectors/reviewers, and the CCB itself. Fifteen out of these 16 organisations document the decisions made by the CCB and their motivations, but only within eleven organisations are these decisions visible via the problem report repository and tracking system.

3.11 Modification Implementation

We have checked whether the organisations studied record all the changes made to software systems, whether they implement these changes according to some documentation standard, and whether the organisations inspect/review the changes made.

Recording Changes to Software (see XI: Q.1, Q.1.1): All the organisations studied record the changes made to the system. These changes are either visible in the problem report repository and tracking system, release documentation, or in the code - in its comment part.

The granularity level of recording the changes is usually on a module level (the identification of a module that has been changed). Finer granularity level such as the identification of the lines being changed due to a particular problem can be achieved manually either via a *"diff"* command in a version management tool or by reading comments in the modules identifying the lines that have been changed.

Rules for Documentation Standard (see XI: Q.3): Fifteen out of 19 organisations have defined rules for how to write software code and documentation. The interviewees from these organisations claim that they follow these rules. The remaining four organisations have not defined such rules at all.

Inspecting/Reviewing Changes (adapted from IX: Q.7): The practice of inspecting/reviewing changes is followed by the same organisations as the practice of inspecting/reviewing suggestions for problem solutions: 15 out of 19 organisations conduct inspections and/or informal reviews on all changes made; four do not.

4 Conclusions

Problem management process is one of the most important processes within software engineering. Its main role is to manage software problems in a formal and disciplined way. Its other role is to help us look back in time and analyse our earlier processes (either development or other maintenance process) in order to identify their weak and strong points. These points are an important feedback for continuous process assessment and improvement, and for piloting innovations exploiting the best engineering practices [1]. For this reason, we need to achieve maximal visibility into problem management process and all other development and maintenance processes.

The problem management process visibility equips us with data providing an important basis for assessing product quality and reliability, crucial for continuous process analysis and improvement, and essential for defect prevention. Good insight into the process helps us make sound decisions. Mature problem management process is a prerequisite for achieving CMM Level 5 [1].

CM^3: Problem Management is a very detailed problem management process model. It has been developed at ABB and evaluated for its industrial relevance within 17 non-ABB organisations. In this paper, we have presented our evaluation results of *CM^3: Problem Management*.

Our evaluation results show, that all the process steps of our model have been either explicitly or implicitly implemented by the majority of the organisations

studied. This proves that *CM³: Problem Management* is realistic and down-to-earth, and correctly represents current industrial practice. The state of implementing our process steps varies across the organisations studied. Some of them implemented most of our process steps, whereas other have hardly defined any problem management process. By delineating the strong and weak points in the industrial processes studied, we hope that we have been able to better visualise the current state of problem management practice within corrective maintenance, and thereby, to provide a basic reference point from which the desperately needed research into software maintenance can proceed.

Acknowledgements

I would like to thank all the software engineers at ABB who co-operated with me personally. They are: Ulf Westblom from ABB Corporate Research; Sari Ebarasi, Margaretha Holmgren, Bengt Kelvinius, Bengt Jönsson, Mats Medin, Ulf Olsson and Stefan Törnqvist from ABB APR; Stefan Forssander, Gunnar Andersson, Tord Fahlgren, Elisabet de Waal, Gunilla Sundelius, Sven-Erik Johansson and Pär Andersson from ABB Robotics, and finally Lars-Olof Tjerngren from ABB Service.

I appreciate the contributions of the practitioners who participated in the evaluation of our model. They are Jan-Eric Claesson from Nexus; Håkan Andersskär from Ericsson Infotech, Leif Thedvall from Postgirot Bank; Jan Lindviken from Ericsson Global IT Services; Birgitta Ervik from Ericsson AB, Claes Ericsson from eumetrix Financial Solutions; Jordanis Caracolias from Ericsson Telecom; Åsa Gustafson from Navigera Business Consulting; Per Foyer from Foyer Consulting; Göran Näsman from On Line; Helena Lindström from Sema Group Infodata; Bo Andersson from WM Data Public Partner; Jorge Amador Monteverde from European Space Agency; Conny Axeus and Per Simonsson from Ericsson Radio Systems; Tore Isacsson from Försvarets Radioanstallt; Jan-Ola Kruger, Kjell Alm and Mats Rundqvist from SAAB Aerospace; and finally, Toni Baknor from Tele2.

Let me thank the Swedish National Board for Industrial and Technical Development (Nutek). This study has been made possible thanks to their recognition of the importance of unifying academia with industry in Sweden. Thanks to its support, the impossible research has been made possible.

Finally, I would like to thank Telefonaktiebolaget LM Ericssons Stiftelse for their financial support allowing us to validate our model within Ericsson Group.

References

1. Carnegie Mellon University, Software Engineering Institute, *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley, 1994.
2. Kajko-Mattsson M, Corrective Maintenance Maturity Model: Problem Management, PhD thesis, ISBN Nr 91-7265-311-6, ISSN 1101-8526, ISRN SU-KTH/DSV/R--01/15, Department of Computer and Systems Sciences (DSV), Stockholm University and Royal Institute of Technology, 2001.

5 Appendix A.1 : Our Questionnaire, Part I

I. Process Definition and Process Visibility

Q.1: Have you defined a model for problem management?
Q.2: Do you divide your problem management process into phases?
Q.2.1: What do they look like?
Q.2.2: Have you defined process models for the following activities: (1) Problem investigation, (2) Problem cause identification, (3) Modification design, (4) Root cause analysis, (5) Modification implementation?
Q.3: Are the problem management phases further divided into activities/tasks?
Q.4: For each problem reporting phase/activity/task:
Q.4.1: Do you record the date and time when the problem report was in this phase/activity/task?
Q.4.2: Do you identify and record additional activities/tasks performed during this phase?
Q.4.3: Do you record the effort and resources of each phase/activity/task?

II. Process Analysis and Control

Q.1: Do you measure the progress? How?
Q.2: Are all problems uniquely identified?
Q.3: How many problems do you describe in one problem report?
Q.4: Do you identify the unique problems?
Q.5: How do you manage unique and duplicate problems?
Q.6: Do you revise the uniqueness of the problem throughout the problem management process?
Q.7: Do you continuously check the correctness of the problem report data?
Q.8: Do you check the quality of the reported data before assigning the problem report to some maintenance team?
Q.9: Do you classify problem reports into external and internal ones?
Q.9.1: If yes, what do you use this classification for?
Q.10: Do you identify a maintenance category for each software problem?
Q.11: Do you revise the maintenance category continuously during the problem management process?
Q.12: What types of maintenance categories do you distinguish?
Q.13: Do you record severity and priority of the problem?
Q.14: Do you separate the submitter's and maintainer's judgement of severity and priority?
Q.15: What do you use these values for?
Q.16: Do you identify the activity during which the problem was encountered?
Q.16.1: If yes, what do you use this for?
Q.17: Do you record the date and time when the problem was reported?
Q.18: Do you compare the plans to the actual results in order to improve the process?

III. Managing Software Products

Q.1: Do you identify the product in which the problem was encountered?
Q.1.1: Do you identify the product and its release ID in which the problem was encountered?
Q.1.2: Do you identify the product component/function in which the problem was encountered?
Q.1.3: What is the lowest granularity level in which you identify the product in which the problem was encountered?
Q.2: Do you identify the environment of the product in which the problem was encountered?
Q.2.1: What do you mean by the product environment?
Q.2.2: Do you consider information on the customer environment when investigating the problem? Does it help in recreating the problem?
Q.3: Do you record the date and time when the problem was encountered?
Q.3.1: If yes, what do you use this value for?
Q.4: Do you designate the version(s) of the software product in which the problem solution will be implemented?
Q.4.1: What criteria do you use when choosing the versions in which the software product will be investigated?
Q.4.2: Do you continuously revise the designation of) the version(s) of the software product in which the problem solution will be implemented?
Q.5: Do you designate (an) appropriate version(s) of the software product in which the problem will be investigated?
Q.5.1: What criteria do you use when choosing the versions in which the software product will be investigated?
Q.6: Do you ensure the traceability of the modified documentation item to the other (modified and unmodified) documentation items within the system?
Q.6.1: On what granularity level do you achieve this traceability?
Q.7: Do you ensure the traceability of change?
Q.7.1: On what granularity level do you achieve this traceability?

IV. Resource Management

Q.1: Do you identify the problem submitter?
Q.1.1: If it is an external problem submitter, do you have his/her/submitter organisation's identification data easily available?
Q.1.2: Do you often use the submitter's identification data? What for?
Q.1.3: In what situations do you contact problem submitters?
Q.2: Do you allow problem submitters to describe (a) work around(s), that is, a list of a set of steps describing how to avoid the problem?
Q.2.1: What do you use it for?
Q.3: Do you allow problem submitters to describe suggestions for how to solve software problems?
Q.3.1: What do you use it for?
Q.4: Do you record problem reports into the organisation-wide problem report repository and tracking system?
Q.4.1. Do you record all software problems in the repository?
Q.5: Do you assign the problem report to the maintenance team for attendance?
Q.5.1: According to what rules do you choose the maintenance team?
Q.5.2: What is this team responsible for?
Q.6: Exactly what roles are involved in the management of software problems and what exactly do they do?
Q.6.1 Who attends to the reported problems (1) anybody, (2) a specially dedicated maintenance team ?

6 Appendix A.2: Our Questionnaire, Part 2

V. Describing and Reporting Software Problems
Q.1: How do you describe the problem?
Q.1.1: Do you provide a template on how to describe software problems?
Q.1.2: Do you give a general textual description of the problem?
Q.1.3: Do you describe the problem effect(s) and consequence(s)?
Q.1.4: Do you describe the symptoms of the problem?
Q.1.5: How do you use the information on symptoms?
Q.1.6: Do you describe the problem conditions?
Q.1.7: Do you describe how to reproduce the problem?
Q.1.7.1: Do you classify the problem as reproducible or non-reproducible?
Q.1.7.2: Do you indicate the repeatability of the problem (once, several, repeatable)?
Q.1.7.3: Do you describe how to reproduce the problem?
Q.1.8: Do you describe alternative execution path(s) to the problem?
Q.2: Do you attach relevant file(s) for visualising/confirming the problem?
Q.3: Do you identify the type of a problem, for instance, requirement problem, design problem, software code problem, etc.?
Q.3.1: If yes, what does your classification look like?
Q.3.2: What do you use this classification for?
Q.4: Do you identify problems related to the reported problem, if any?
Q.5: Do you revise the software problem report data continuously during the problem management process as you gain more understanding about the problem?
Q.6: Do you submit additional problems encountered during the problem management process?

VI. Problem Investigation
Q.1: Do you study the problem report?
Q.1.1: Do you encounter any problems/difficulties during this step?
Q.2: Do you study the software system?
Q.2.1: Do you spend enough time for studying the software system?
Q.2.2: Do you fully understand the software system?
Q.3: Do you define a set of test cases required for the recreation of some problems or do you just recreate the problem in an ad hoc manner?
Q.3.1: Do you document these test cases?
Q.4: What exactly do you document during problem investigation?
Q.5: What else do you do when investigating problems?

VII. Problem Cause Identification
Q1: Do you study the results of the activity *"Problem Investigation"* in order to get acquainted with (if another Problem Report Engineer) or get reacquainted with the problem?
Q.2: When identifying the problem causes, what exactly do you do?
Q.2.1: Do you study the pertinent documentation items and check their correctness?
Q.2.2: Do you identify and record the documentation items containing the defect(s)?
Q.2.3: On what granularity level do you record the problem causes (component/line level)?
Q.3: Do you classify the identified problem cause(s)/(defect(s))?
Q.4: What do you use this classification for?
Q.5: What exactly do you document during this activity?

VIII. Root Cause Analysis
Q.1: Do you identify the process activities/tasks during which the problem cause (defect) was introduced?
Q.2: Do you identify the root causes for the defect by analysing the process steps during which the problem cause/(defect) was introduced? How do you do it?
Q.3: Do you classify the root cause(s)?
Q.4: Why do you conduct root cause analysis?

IX. Modification Design
Q.1: How many suggestions for problem solution do you make?
Q.2: Do you record all these suggestions?
Q.3: What exactly is recorded: (1) a general description of what a problem solution should look like, (2) a specification of measures to be conducted in order to solve the problem?
Q.4: Do you evaluate the problem solution?
Q.5: What is exactly evaluated: (1) a general description of what a problem solution should look like, (2) a specification of measures to be conducted in order to solve the problem?
Q.6: When evaluating problem solutions, what exactly do you consider (1) modification size, (2) impact of the modification, ripple effect, effect on customer, (3) the effort and resources required for solving the problem, (4) the feasibility of solving the problem, (5) the benefits and drawbacks of solving the problem?
Q.7: Do you inspect/review the modification designs?
Q.8: Who does the inspection/reviewing?
Q.9: Do you make an implementation plan for each problem solution?
Q.10: What exactly do you plan?
Q.11: When at the earliest do you start designing problem solutions?
Q.12: Do you evaluate the early preliminary problem solutions?

X. Modification Decision (Stage 4)
Q.1: Do you report the results of the major process activity to some authority?
Q.2: Do you make decisions whether to proceed with the problem resolution after each major process activity?
Q.3: Do you have a Change Control Board (CCB)?
Q.4: What kinds of decisions does CCB make?
Q.5: When making decisions on the choice of modification designs, what exactly does the CCB do?
Q.6: Do you record all the decisions made (1) after each major process phase, (2) decisions made for all solutions?

XI. Modification Implementation (Stage 5)
Q.1: Do you record all the changes made to the software product?
Q.1.1: On what granularity level?
Q.2: Do you unit test these changes?
Q.3: Do you modify the product according to some documentation standard?
Q.4: For each change, do you record the following: (1) reason for the changes, (2) modification size, (3)effort and resources, (4) ripple effect, (5) feasibility, (6) drawbacks and benefits?
Q4.1: Do you record this information for the whole modification effort or for each individual measure specified in problem solution?