Software maintenance: an analysis of industrial needs and constraints

Marc Haziza, Jean-François Voidrot, Jean-Pierre Queille, Lech Pofelski, Sandrine Blazy

Software Maintenance: An Analysis of Industrial Needs and Constraints

M. Haziza, J.F. Voidrot
Matra Marconi Space France
31, rue des Cosmonautes
31077 Toulouse Cedex, France

E. Minor, L. Pofelski
Cap Gemini Innovation
7, chemin du Vieux Chêne
38240 Meylan, France

S. Blazy
Electricité de France
1, avenue du Général de Gaulle
92141 Clamart Cedex, France

Abstract

A prerequisite to improving the effectiveness of software maintenance activities is the attainment of a deep understanding of existing industrial maintenance practices. This paper reports the results of a series of case studies which were conducted at different industrial sites in the framework of the ESF/EPSOM project. The approach taken in the case studies was to directly contact software maintainers and obtain their own view of their activity, mainly through the use of interactive methods based on group work. This approach is intended to complement statistical studies which can be found in the literature, by bringing the perspective of the maintainers based on their experience. The aim of these studies has been to gain a better understanding of maintenance needs and constraints, and to highlight directions which could lead to improvements in the quality and efficiency of maintenance activities.

Introduction

Improvement in the quality and efficiency of maintenance activities requires a deep understanding of current maintenance practices. Considerable research effort has been directed to studying the practice of maintenance, and identifying important influences. The classic studies of Lientz and Swanson [8], [9], [10] laid the groundwork for this research by presenting broad surveys of maintenance activity based on questionnaires filled out by maintenance managers. A recent follow up by Nosek and Palvia [12] used essentially the same questionnaire to give these studies a longitudinal dimension of ten years. A different approach can be seen in research based on case studies. The studies of Calow [2] and Abran and Nguyenkim [1] both gather data on large maintenance activities for periods of several months to two years. The data is then analysed statistically to determine a profile of maintenance activity, and to identify problem areas.

As an early part of the ESF/EPSOM[1] project, the partners wanted to perform a study of maintenance practices in the types of industrial settings which are of greatest interest to their companies and clients. They decided to employ a case study approach, but to give the case studies a new dimension, by concentrating on the maintainers themselves and getting their own views of their activity. This approach is similar to the field studies of the software design process described by Curtis, Krasner and Iscoe [3]. It was felt that it could be complementary to the purely statistical data, providing an interpretation based on the maintainers experience.

This paper is a report of the case studies. First, the methods used to perform the studies are described in section 1. Next, an overview of the different industrial settings chosen for the studies is presented in section 2. A synthesis of the study results is presented in section 3, with emphasis on needs and constraints expressed by maintainers. Section 4 briefly discusses the applicability of the results to other maintenance settings, and finally, section 5 presents the partners views on how the study results can help direct future work.

1. Methods

The choice of methods for conducting the case studies was motivated by the desire to engage practising software maintainers in as large and active a role as possible. Beyond this basic consideration, there were differences in the methods employed by each research partner stemming

[1] *EPSOM (European Platform for SOftware Maintenance) is a subproject of the Eurêka Software Factory project (ESF). It aims at investigating the concept of integrated environments for software maintenance, following two directions: the definition of generic models of software maintenance processes and the implementation of a generic environment for software maintenance. The project is partially funded by the French authorities, under the Eurêka programme. Current partners in the subproject are Matra Marconi Space France (MMS), Cap Gemini Innovation (CGInn) and Electricité de France (EDF).*

0-8186-2980-0/92 $03.00 © 1992 IEEE

from differences in the case study settings such as size of maintenance team, access to maintenance team, and organizational relationship between the research team and the maintenance team.

Both MMS and CGInn chose methods which were centered around interactive group work. This approach was considered the most interesting way to achieve active maintainer participation for medium to large sized teams. Both partners supplemented group work with structured interviews or questionnaires in order to determine the overall maintenance context for each case study.

EDF chose a method based on in-depth questionnaires. This approach was adopted after an initial survey revealed that maintenance at their site concerned a large variety of software and tended to be carried out by small teams, often devoting only part of their time to maintenance. In such a setting it was decided that the most interesting results could be obtained by finding a group of willing volunteers to answer an in-depth questionnaire, containing a large percentage of essay type questions.

The following sections discuss in greater detail the methods used for the case studies.

1.1. Determining context

For each case study, information was gathered which permitted the overall context of the maintenance activity to be determined. This information was useful within a study for explaining why certain issues were important to the maintenance team. It also provided criteria for making comparisons between case studies, based on the knowledge that there were similarities or differences in their contexts. The context information included [5]:

- profile of organization which the maintenance team supports: site, nature of activity, economic context
- profile of maintenance organization itself: size, management structure, professional and educational background of the maintainers
- main characteristics of the maintained system: hardware types, software size and languages, commercial packages used, documentation size and medium, key project dates, development methods used
- technical support environment: methods, special tools, types of computer systems to perform maintenance, use of configuration management

This information was obtained through structured interviews or questionnaires.

1.2. Group work

For MMS and CGInn, group work formed the core of the case studies. The motivation for employing group work was to engage maintainers and obtain their ideas; perhaps ideas which were unanticipated. For the group work, the partners adopted three techniques which are used extensively in the framework of "quality circles": brainstorming, voting, and Ishikawa diagrams.

Brainstorming techniques

Brainstorming was the first phase of the group work. The goal of brainstorming is to gather as many ideas as possible [11]. Each idea is formulated by a single person, in a few words, on a "post-it" and placed on a board where everyone can see it. It is essential during the session to maintain a non-critical atmosphere, encourage the production of ideas, and assure the participation of all members. Discussion of the quality of ideas is prohibited. The only discussion which should occur is that required for all members to be sure they understand each idea. The sessions lasted until no one had a new idea to contribute, typically 45 minutes to 1.5 hours.

In the brainstorming sessions, the participants were asked to identify all the factors which affect software maintenance, based on their own experience.

Voting techniques

The goal of voting is to allow the maintainer to make judgements about the relative importance of the ideas generated in the brainstorming. Voting is done individually, and the results are combined to produce a group result. Different techniques may be used, for example rank order the N most important ideas from 1 to N, or place all of the ideas in one of N classes depending on their importance. Voting may be iterative, by decreasing and then relaxing the number of selected ideas at each step.

MMS performed voting immediately after the brainstorming, using a rank ordering technique. CGInn used a descendant weighted technique, performed after the construction of the Ishikawa diagram.

Ishikawa diagrams

Ishikawa diagrams [7], also called cause-effect or "fish-bone" diagrams, are used to structure ideas into hierarchies, and result in a simple graphical representation which, in a group work setting, is a consensus view.

An Ishikawa diagram is constructed by placing a bold horizontal arrow, pointing to an effect to be analysed. The principal causes impacting the effect are then added as diagonal arrows pointing to the central arrow. Subordinate

clusters of arrows are then added to represent subordinate levels of causes.

In the case studies, the maintenance teams were presented with Ishikawa diagrams where the effect and the principal causes were already defined. They were then asked to place the brainstorming ideas on the principal diagonals, and to develop sub-structures where appropriate. The resulting diagram, along with the results from voting, allowed quantitative weights to be assigned to axes and sub-axes, and relative importance of ideas to be assessed.

Figure 1 shows an example of an Ishikawa diagram, in which the six principal cause axes were chosen to cover the main classes of technical, strategic, and socioeconomic factors which impact maintenance quality. The diagram shows most of the factors which are generally identified throughout the literature [4].

For its case studies, MMS adopted the first level structure of figure 1, with six classes. In addition, MMS used a second type of diagram, with main diagonals representing process steps, which allowed maintainers to identify how much effort they devoted to successive steps of a typical maintenance process.

For its case studies, CGInn adapted the frequently used "4M" Ishikawa diagram to the context of software maintenance, obtaining the four principal causes: Men, Methods, Maintained product, and Maintenance tools.

1.3. Questionnaires

Due to the large variety of software systems and maintenance sites concerned by the study, EDF chose to send questionnaires to selected people. The questionnaire was broken down to 4 parts: identification of the maintenance team, typology of the maintained system, maintenance process, and maintenance problematics.

2. Selection of case studies

The selection of case study targets was motivated by the desire to cover a wide scope of industrial maintenance activities, and many different types of maintenance teams. In addition, the goal was to perform in-depth studies, rather

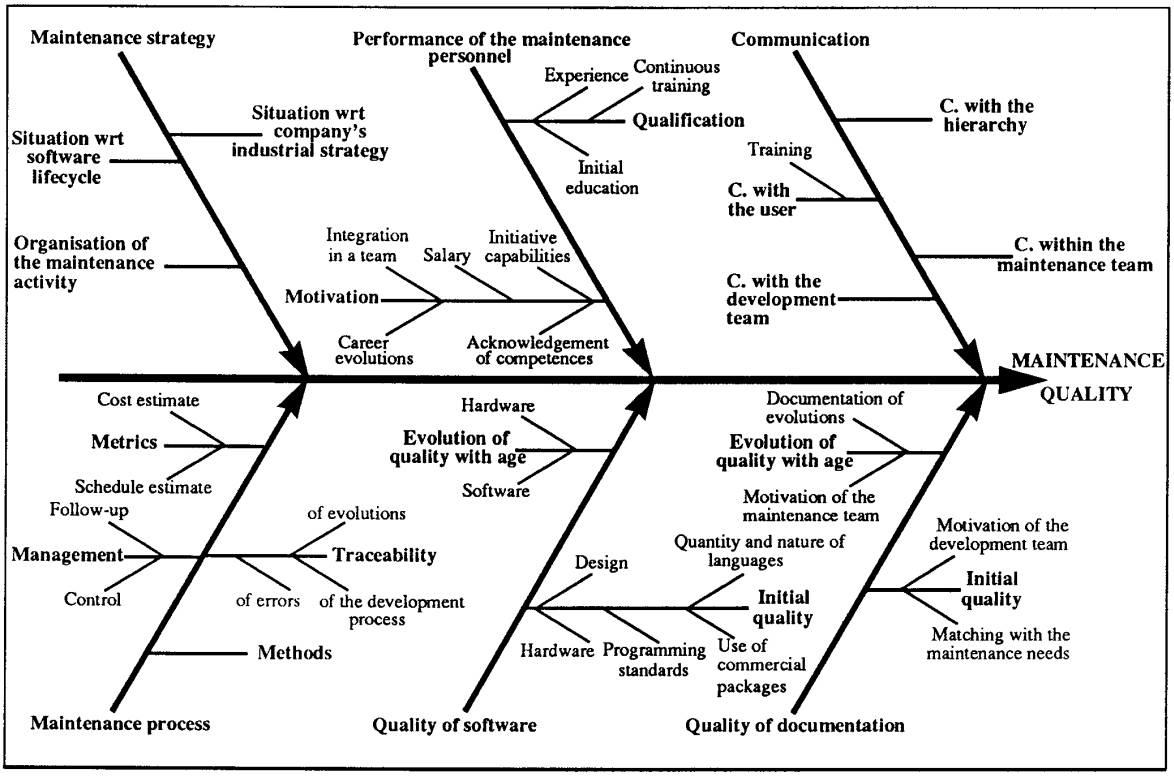

Figure 1. Main quality factors for software maintenance

than cover a larger set of cases but treat them more superficially. Due to privacy issues concerning some of the studies, and because of the geographic dispersion of the three partners, each partner was responsible for choosing their own case study targets and for gathering the data. The partners worked together before conducting the studies in order to establish the broad outlines, and then again after the data gathering in order to analyse the results.

Matra Marconi Space selected their case study targets from within their own organization. MMS has been the prime contractor for numerous space system projects including the TELECOM 1 and 2 communication satellites, the SPOT 1, 2, and 3 earth observation satellites, and components of the ARIANE and HERMES space transportation systems. These projects have involved MMS in the development and maintenance of many large software systems. Thus there was a rich in-house field for selecting case study targets, and a high degree of company motivation and support for pursuing such studies.

Four case studies were selected, all from the "maintenance of ground systems" department. The cases were chosen according to criteria such as representativeness with regard to space activity, maintenance team profile, economic context, characteristics of the maintained product, and maintenance history. The studies covered a range of real time software systems for spacecraft assembly, integration, and test, or operation center. These applications implement a wide range of machines, languages (incl. Fortran and assembler) and methods and must comply with the stringent constraints of the space activity.

Cap Gemini Innovation is the technology transfer branch of the Cap Gemini Sogeti group, the largest software services company in Europe. They selected their case study targets from existing or potential clients. These clients all have a large interest in analysing and improving their maintenance activities. Four case studies were selected:

1) maintenance of CASE tools written in C and FORTRAN,
2) maintenance of real-time assembly language programs for the nuclear power industry,
3) maintenance of telecom and teletext applications in assembler and Pascal,
4) maintenance of a large COBOL transaction system,

The first three case studies were carried out in France, and the fourth at a company in England.

Electricité de France is the national electric utility for France. Their research and development center at Clamart has been developing large scientific applications in FORTRAN for decades. These applications include network planning, electric power dispatching, nuclear plant refuelling, and simulation of the behaviour of nuclear plant core. There is an interest in reducing the time and effort devoted to maintenance for these applications, and so EDF chose to conduct their case studies at their Clamart site.

EDF obtained twenty volunteers to complete the in-depth questionnaire. This allowed them to gather results about ten applications written in FORTRAN, varying from code developed and still maintained by the same small team to code developed in a CASE environment with quality assurance considerations.

3. Synthesis of results

Group work and analysis of questionnaires have provided a better understanding of how maintenance teams perceived their needs in terms of organization, methods and means (tools, logistics). These different classes of needs are presented hereafter.

3.1. Organizational issues

All teams stressed the importance of strong organizational support for maintenance. There was variation in terms of actual support, with some teams expressing satisfaction with their organization, and others expressing that their role was undervalued. A neglect of maintenance by the organization goes along with scarce management and assignment of minimum resources, and this is recognized by the team. A significant effort should be made to improve the recognition of maintenance at management level. This may lead to a structuring of the maintenance activity in a department with its own strategy and R&D policy.

Within the maintenance team, teamwork is considered important. Situations where a unique person is in charge of maintenance for a system should be avoided, and the management should encourage technical exchanges between maintainers.

Maintainers should not be diverted from their technical work to perform activities such as secretarial work or configuration management.

For some teams a very specific problem was noted: the requirement to perform maintenance at remote sites, at difficult hours, under pressure. This type of maintenance was uniformly deemed to be unsatisfactory and insufficiently compensated.

3.2. Individual performance of the maintainer

Due to the artisanal stage of maintenance, and the fact that corrective maintenance deals with problems sometimes subtle and hard to reproduce and understand, the aspect of individual contribution or profile of the maintainer is very strong. The notion of maintainer's qualification is twofold:

- technical experience,
- knowledge of the maintained system and its environment.

Technical skills, but also profile considerations such as motivation and a "strong" character are judged very important for the efficiency of the maintainer. An experience in the development of software systems is also seen as improving the performance of the maintenance staff. Bad experiences with beginners or persons not sufficiently motivated by the maintenance activity were reported. For all those reasons, assignment to a maintenance function should result from a free choice. Opportunities should be given to the maintainer to work on products implementing advanced technologies, in order to limit the obsolescence of knowledge. It should also be possible for the maintainer to evolve towards other fields or other types of activities.

The need has been expressed for a better knowledge of the "real world" from which the maintained system is an artifact, especially in the context of space systems or scientific applications. This need includes better understanding of the vocabulary of the domain, of the nature and purpose of the objects handled by the system.

There is also a need for better system insight, clarifying how the maintained system is embedded into a higher level system with its proper mission, logic and constraints (e.g. in the space context, a real time monitoring system in a control centre, or the control centre itself as an element of a control loop, including other earth stations, ground networks, other in-orbit components such as data relay satellites).

Maintainers need to better understand the way the product is used in the operational environment. Maintainers must share a common knowledge of the operational context with the users. This is especially true for control systems where the knowledge of the operational environment is a sine qua non condition for problem understanding.

The participation of maintainers to the specification and integration of the systems they will have to maintain has been recommended.

Maintenance teams generally experience high pressure from users. Ways to lower this pressure should be looked for. This may involve increasing the autonomy of users, filtering users requests, discharging maintainers from repetitive tasks with low added value, and providing appropriate support so as to shorten the maintenance cycle.

3.3. Knowledge management and communication

A major need has been expressed by almost all maintainers for better means to capitalize and transfer knowledge on the products and processes both during the development and the maintenance phases. This need has sometimes been expressed as a need for communication between the development team and the maintenance team, or within the maintenance team itself.

The initial transfer of knowledge between the development and the maintenance teams should be as complete as possible.

As the lifetime of software products is often very long, there is also a need for long-term conservation, dissemination and refinement of expertise from one maintainer to another. Training and better documentation are seen as ways to improve the integration of new maintainers into existing teams.

The possibility of getting help from developers during the maintenance phase is another way to achieve the transfer of knowledge between the development and the maintenance teams. This implies that developers are still present, that they are made available to support the maintenance team and that this support role is clearly reflected at the managerial level.

Maintenance teams have a strong interest in maintaining a history of the development trade-offs and their justification. Solutions to capture the "good" information from the development team (including the meaning of codes and mnemonics used during the development) should be defined. Knowledge about the different kinds of links existing between and within the software product and the documentation should be preserved.

On the users' side, a means should be found to better preserve their knowledge and to improve their self-sufficiency. A relationship grounded on mutual confidence should be established between the users and the maintenance team. This is a condition to a better understanding of the users' point of view and problems by the maintenance team, or conversely to a better understanding of the socio-economical context of the maintenance activity by the users. Some maintainers have also suggested that users' training sessions should be planned on user request, in order to respond to specific needs.

Finally, communication with the hierarchy should be improved. This includes a means for the management to

better understand the nature of the work and to better evaluate the resources needed for maintenance interventions.

3.4. Maintenance process and support environment

Formalisation of the software maintenance process is generally much poorer than of the development process. In some situations there may even be no formalisation at all. On the other hand, well defined controls of the maintenance process have been implemented on some sites to comply with high reliability requirements.

Maintainers generally judge that a better formalisation of the maintenance process and a better knowledge of this process by the maintenance staff would greatly improve the efficiency of the activity.

There is a consensus about the need for very well formalised procedures, specially for corrective maintenance.

The maintenance process should be precisely and completely defined. The processing of problem reports and relationships with clients should be fully formalised by procedures. Maintainers should have easy access to this knowledge.

The need for a better support environment has several facets:

- Management tools are needed, in order to plan the maintenance activity, to track maintenance tasks, to keep the history of problems and solutions, and to organize and search this history.
- Technical activities should be better defined and tools are needed to support these activities. Needs are expressed for nearly all the steps involved in maintenance interventions:
  - diagnosis level (e.g. discrimination between hardware and software failures),
  - static and dynamic analysis of code (call graphs, localisation of pertinent information, debugging),
  - impact analysis (mastering impacts not only on code, but also on documentation, on performances),
  - test generation,
  - updating and execution,
  - regression analysis.

  Though these needs are sometimes supposedly covered by commercial tools, it appears that these tools are seldom adapted to the operational context of maintenance (languages, operating systems, machines), specially in the case of "old" systems.
- Needs are also expressed in the domain of version and configuration management: integration of configuration management on the maintenance and the operational sites, and multi-site and multi-machine configuration management.

3.5. Quality of the maintained product

The use of development methods, of CASE tools, and compliance with programming guidelines during the development and the maintenance phases are always considered as having a strong impact on the efficiency of maintenance.

The existence of appropriate development documentation is generally seen as a major factor in good maintenance. This documentation must have been designed with maintenance in mind, because maintenance teams have specific needs which generally are not well covered by standard documentation produced during the development. Furthermore, the volume of documentation issued from the development is generally huge, which renders difficult the retrieval of the appropriate information, especially for newcomers. So there is not only a need for a better structure and content of documentation, but also for tools which help to search and navigate through documentation.

However, some maintainers have expressed a different point of view: quality of code is preferred to documentation, all the more if the volume of documentation is large, which induces difficulties in using and updating it. In that case, good quality source code is sometimes considered preferable in order to achieve a good comprehension of the software.

The lack of quality assurance during development cannot be bypassed at the maintenance stage. Unresolved difficulties, unit and integration testing done without care, and maintenance not taken into account during development cause considerable problems during the maintenance phase.

The use of modern languages for development is needed to improve the quality of code, and thus to ease maintenance. For example, the use of FORTRAN in large scientific software makes maintenance harder, as FORTRAN does not facilitate structured programming. It lacks structured types and dynamic memory allocation. This implies trade-offs from the development teams, which may impact on code clarity.

4. Applicability of the results

The case study partners recognize that care must be taken in claiming a wide applicability for the results. A

fairly limited number of maintenance teams were studied, all but one of which were located in France. The Ishikawa diagrams are qualitative in nature, and they differ from one team to the next. The questionnaires are also qualitative, consisting largely of essays expressing maintainer's ideas. The case study targets were not chosen to be a representative sample, but simply to be of specific, immediate interest to the partners. For all of these reasons, the study must be viewed as a snapshot of a particular set of maintenance activities.

Nevertheless, the partners believe that many of the needs and constraints expressed by maintainers in this study are of general concern. Several of the case study targets, particularly those for space, nuclear power, and telecom, are typical of large maintenance segments. In the one cross-cultural study, dealing with a maintenance team in England, the partners discovered that "we all speak the same language" (literally). The Ishikawa diagrams differ in structure and vocabulary, but many of the underlying ideas recur. The voting technique, although not truly quantitative, does permit a relative ordering of ideas, and the key ideas appeared as important in most of the studies. These same ideas were often brought out in the questionnaires. For these reasons, the partners would expect to see similar results if the studies were conducted with many more teams, in different countries.

The selection of cases has privileged "old" projects, or projects still on-going but started long ago. However, the case studies reported here are representative of the constraints of complex industrial applications, and it can be anticipated that even modern applications developed with state-of-the-art techniques and methods will experience the same kind of difficulties by the end of their lifetime. The space domain provides a good illustration of this kind of situation. The lifetime of a space system is very long. There can be more than 20 years between the design of a system and the end of its operational life (40 years for some systems currently designed, like the HERMES/ COLUMBUS infrastructure). As a consequence, even if development methods are up-to-date at the time of design, they will very probably become obsolete during the maintenance phase. Such lifetimes also imply a high personnel turnover, which renders difficult the preservation and dissemination of knowledge.

5. Lessons for the future

The weighting of Ishikawa diagrams has allowed us to obtain a first indication of the relative importance of each class of maintenance quality factors. This approximate quantification of needs is sufficient to prioritize, in the selected industrial context, the different types of actions that could be undertaken to improve the quality of maintenance. The two main directions identified through this process are:

1) the transfer, preservation and maintenance of knowledge,

2) the mastering of the maintenance process.

A number of complementary actions that are judged important in the perspective of improving the maintenance quality were also proposed.

5.1. Acquisition, transfer and maintenance of knowledge

This issue can be seen from different points of view depending on the context of the project studied. It is sometimes expressed as a need for development documentation of better quality, or a need for highly experienced and skilled maintainers, who are able to draw from their knowledge to solve problems, or a need for improved communication with the development team, the users, and within the maintenance team. These are different aspects of the same problem: knowledge of the maintained system and its environment must be preserved and accessed during the whole life-cycle of the system.

In this perspective, a first step is to identify the "good" information for the maintainer. It may correspond to:

- information already present in the development documentation, or which can be automatically extracted from code with appropriate tools,
- design trade-offs made during the development phase, and which cannot be found in the reference documentation,
- information concerning the history of development and maintenance,

This may also be information about the "real world" from which the software system is an artifact, about the way it is integrated and used in the operational context, about the vocabulary of the domain.

Most of this information evolves during the lifetime of the system.

The relevance of this information is also closely dependent on the maintainer's profile (education, experience).

The most appropriate media for the transfer of the various types of information between the development and the maintenance teams, and for the management of this information during the whole maintenance phase should be identified. Information support may include maintenance plans, databases, history documents, knowledge-based

systems, documentation models (SGML is eagerly waited), CASE tools (e.g. extraction of data flows).

These different information supports should be integrated into a coherent information system providing a single point of access to the maintainer, offering modern utilization features such as hypertext-like and browsing tools.

The identified solutions may also lead to a reconsideration of the structure and content of the development documentation currently required by the development standards.

5.2. Modelling and control of maintenance activities

Nearly all maintainers expressed a need for a better mastering of the maintenance process for both managerial and technical activities. Process modelling is the initial step in this direction. Process modelling is defined as the detailed analysis and modelling of maintenance activities in order to better understand the process (descriptive point of view), to better guide it (incitative point of view), to enforce it (prescriptive point of view), and to better tame it with regard to the cost / delay aspects.

For this last objective, those models need to be calibrated. Information about the costs and resources needed by each class of activities has to be derived from past and on-going projects. The observed costs may also be used to improve the maintenance process. The calibrated model can then be instantiated and adapted to the particular context of a given project, implemented using an appropriate process modelling tool, and connected to planning and monitoring tools to transform it into an operational tool for the management of maintenance activities (planning and follow-up of activities, resource allocation).

5.3. Other directions

Several other directions have been identified in order to improve the maintenance quality.

Maintenance needs tools to perform technical activities such as code analysis, impact analysis, regression testing. Several tools are currently available on the market place. They are either development tools which are naturally usable in the context of maintenance activities (code analysers, debuggers, or quality tools), or tools specific to maintenance activities (e.g. reverse engineering tools). These tools generally need some level of customization, and their applicability to the selected industrial contexts has to be studied case by case. Some other needs are not covered by currently available tools.

Support to first level diagnosis is another direction in which effort must be placed. Means must be found to reduce problems related to diagnosis on remote sites, to improve the users' self-sufficiency, to decrease maintainers' workload by automatically processing the most common problems.

Configuration management is generally well mastered by maintenance teams. However, some specific post-delivery problems are still open. They mainly concern the consistency of the configuration management between the operational and maintenance sites, multi-site / multi-machine configuration management, or configuration management of a family of systems.

Finally, an effort should be made towards giving developers and users a better awareness of maintenance needs and constraints. This should improve the efficiency of maintenance teams. The problem goes beyond the strict scope of maintenance, and deals with the improvement of the overall quality of the production tool.

6. Conclusion

The case studies reported in this paper have allowed a better understanding of maintenance needs and constraints in selected domains of the software industry. The results obtained tend to confirm the main preoccupations of the maintenance community, with an emphasis on two types of needs which appear crucial in the domains of activity of the partners, namely:

1) the transfer, preservation and maintenance of knowledge,

2) the mastering of the maintenance process.

These two major needs confirm the initial orientations of the EPSOM project:

- the concept of "traceability platform", which attempts to provide a support for all kinds of relations existing between software or documentation components (including browsing and management tools),
- process modelling activities, including the definition of the process models of maintenance [6], and their implementation using the available technology.

Other needs that are judged important in the perspective of improving maintenance quality have also been identified:

- tools supporting technical maintenance activities,
- support for first level diagnosis,
- more powerful configuration management tools,

- and a better awareness from the developers and users of the needs and constraints of maintenance.

Even if nothing really new has emerged from these case studies, the partners feel that they were very profitable for the continuation of the project, as they confirmed and deepened their understanding of the actual needs, and their feeling that software maintenance support is something worth working on.

For that purpose, one in-the-field study is better than ten literature surveys!

References

[1] A. Abran and H. Nguyenkim: "Analysis of maintenance work categories through measurement", IEEE Conference on Software Maintenance, Sorrento, Italy, October 1991.

[2] H. Calow: "Maintenance productivity factors - A case study", IEEE Conference on Software Maintenance, Sorrento, Italy, October 1991.

[3] B. Curtis, H. Krasner, N. Iscoe: "A field study of the software design process for large systems", Communications of the ACM, Vol.31, No.11, 1988.

[4] EPSOM Team: "State of the art in software maintenance", Deliverable D1.1, v. 1.0, ESF/EPSOM project, August 1991.

[5] EPSOM Team: "Software maintenance: Industrial needs and constraints", Deliverable D1.2, v. 1.0, ESF/EPSOM project, December 1991.

[6] D.R. Harjani, J.P. Queille: "A process model for the maintenance of large space systems software", IEEE Conference on Software Maintenance, Orlando, USA, November 1992.

[7] K. Ishikawa: "What is quality control? The japanese way", Prentice Hall, 1985.

[8] B.P. Lientz, E.B. Swanson, and Tompkins: "Characteristics of application software maintenance", Communications of the ACM, Vol.21, No.6, 1978.

[9] B. P. Lientz and E.B. Swanson: "Software maintenance management", Addison-Wesley, Reading, MA, 1980.

[10] B.P. Lientz and E.B. Swanson: "Problems in application software maintenance", Communications of the ACM, Vol.24, No.11, 1981.

[11] K.L. MacGraw and K. Harbison-Briggs: "Knowledge acquisition: principles and guidelines", Prentice Hall, 1989.

[12] T. Nosek and P. Palvia: "Software maintenance management: changes in the last decade", Journal of Software Maintenance, Vol.2, No.3, 1990.