A Guide for Software Maintenance Evaluation: Experience Report.

Véronique Narat, Arthur Vila

Electricité de France,Direction des Etudes et Recherches
Service Informatique et Mathématiques Appliquées
1, avenue du Général de Gaulle, 92141 Clamart, France.

Abstract. This paper presents an experience report carried out at Electricité de France, the French electrical power company. It shows how applied research activities on the software maintenance problems are set up within a quality and software engineering division. It then particularly focuses on an experience aiming at assisting software maintenance managers to evaluate their maintenance organisation. This experience is based on a guide that provides questions to ask a software maintenance team, and the way to evaluate the answers and conclude about first steps towards maintenance improvement.

1 The Software Engineering Context at Electricité de France

1.1 Activities and people

Electricité de France (EDF) is the French national company that produces, distributes and provides electricity to the whole country. As such, the EDF has to deal with an extensible amount of varied computerized applications: from data processing applications written in Cobol to scientific applications written in Fortran.

The Research and Development Directorate of Electricité de France (EDF-DER) is in charge of numerous software products used to carry out studies in many technical fields, like mechanics, hydraulics, electric transmissions, or other fields of application. The 1500 Research Engineers in charge of these domains are people of great knowledge in their application field, but are scarcely software engineers first. For them, computers and software are tools, and algorithms are firstly translations of equations. Nevertheless, software production is an increasing part of their activity, because of the continuous improvement of mathematical models and because of the moving computer technology.

Most of the codes are long life software, constantly modified to incorporate improved models of the domain they describe. They are usually written in FORTRAN, even if more and more parts of codes are now written in C or C++ programming languages (especially pre-processors and post-processors). These options are consistent with the computers we have, which are mainly CRAY and IBM mainframes, reached through over 1000 Unix workstations interconnected by FDDI and LAN networks.

1.2 Overview of the Quality and Software Engineering Team

Software Engineering has been a concern for many years at EDF-DER, and a specific team (the Quality and Software Engineering Team) of 13 is in charge of theoritical and practical software engineering related problems. Their role is not to impose software engineering techniques, ideas or tools but rather to assist, give advice and propose solutions to software engineering problems. EDF engineers may ask assistance to the QSE team but they may also ask to other experts inside or outside the company.

In order to be competitive in front of these other experts, QSE has divided its main activities in two parts:

• assistance on projects, either on methodological or on technical aspects,
• theoritical studies based on state-of-the-art issues that have to be adapted to EDF's specific problems.

The four domains in which the Quality and Software Engineering Team is currently working are the following:

• Quality Assurance domain, where the QA manual of EDF-DER is produced as well as guides for the assistance on QA problems and lectures on the subject; and where software code and projects audits are done on request, on the QA point of view.
• Software Development domain, where a CASE tool dedicated to Fortran projects and supporting a methodological approach is provided to projects under development. Lectures on software engineering subjects are also given.
• Information System domain, where a strong support on project organisation and on the analysis and design phases is given and theoritical studies on object-oriented analysis methods are in progress.
• Software maintenance domain, where the problems related to software evolutions are studied, especially on a methodological point of view, in close relationship with the QA and software development domains.

2. The software maintenance group setting-up

Within the QSE team, the software maintenance group is the youngest; it started four years ago, this means that software maintenance has been recently considered as an issue to be studied on its own at EDF-DER. Due to the large number of problems the subject covers and the limited resources we have to carry out these problems, we decided first to analyse the main requirements of EDF-DER in software maintenance of scientific applications, in order to decide the priority domains to work on.

2.1 Requirement analysis phase

Two different ways of working on the requirement analysis phase have been investigated: prototype design and interviews or questionnaires campaign.

We built small tool prototypes with the help of colleagues who agreed to be more involved, in order to highlight specific needs that were not met by market tools and also to demonstrate feasability or difficulty of some of the requirements. This experience showed first the difficulty to provide tools which meet specific needs, but also the necessity to provide such tools in order to establish a dialog between software maintainers and QSE. Talking about a tool is more attractive than just talking in theory and it gives a basis to understand each other.

In parallel, we also sent questionnaires to a large number of project maintainers. These questionnaires covered several aspects of software maintenance: organisation (definition of the roles within the project), method (which method was used during the development, is it still in use), tool (which tools are used), programming rules (do they exist, are they described formally), documentation (which documentation is maintained and how). After a first questionnaire analysis step, we refined this analysis by interviews when necessary, with people who agreed to spend half a day to answer specific questions.

2.2 The software maintenance domains studied

These two ways of defining the software maintenance requirements in a scientific domain led to the exhibition of four main domains to investigate in priority:
1• software maintenance tools,
2• software maintenance documentation,
3• maintenance of software developement methods and tools used,
4• software maintenance organisation.

Following these results, four studies began, each of which dealing with one of those points.

• The first study aims at knowing software maintenance tools, from debuggers to configuration management tools, with a particular attention put on code comprehension tools.

• The second point we are working on concerns the identification of the useful software maintenance documentation.

• The third point is strongly connected to the work done by the software development group into QSE and consists in showing how a CASE tool used during development evolves in maintenance.

• The last point comes from the observation that many software maintenance projects suffer from a lack of organisation. It is usually rather easy to come to this conclusion but more difficult to find the way to cure the situation. Our first work is then to provide ways to find where to put efforts on first improvements. Our approach is described in chapter 3.

3. An approach for software maintenance organisation evaluation

3.1. What are the kind of problems within a software maintenance organisation?

Often time, when all the actors of a project agree to blame its maintenance, no one points out the same responsible cause. If customers tend to claim for quicker fixing and constant evolutions, maintainers incriminate both their managers who seem not to realise the actual technical problems, and the customers who never know what they want, or always want more. Managers who have to balance between customers complaints and detailed technical argumentations from their team must become a referee without always knowing the rules of the game they are all playing. In order to clarify this kind of situation, our approach is supposed to help the project manager analyse the situation by giving him an overview of everyone's point of view.

3.2. The approach

General description: The approach we took is based on the idea that during the software maintenance phase of a project, problems encountered can generally be divided into three classes:
• communication problems,
• techniques problems,
• strategic problems.

Communication problems occur when the role of each actor is not clearly identified. Our approach gives a framework to show the information flow between all the actors.

Techniques problems are due to shortcomings in the maintenance structure. The approach helps settle technical answers to these problems.

Finally, having a better idea of how strategic the project is will assist the decision making on the project maintenance.

The process: Our approach aims at analysing the main class of problems on a project, in order to find what to improve first. It is based on a three-steps process, composed of a questionnaire and a guide to analyse the answers, as described in figure1.

The analysis is supposed to be carried out by the manager of the project currently being maintained. In the first step, the manager sends the questionnaire to all the actors of the maintenance. In the second step, all the answers are collected and analysed with the help of our guide. Finally, a diagnosis is made on the first way where efforts should be put to improve the maintenance.

• *The questionnaire*
The questionnaire (see appendix A) is deliberately short, to be quickly answered. It is divided into three parts of five questions each, corresponding to the three topics (communication, techniques, strategy) we address. It is sent to all the actors of the maintenance in order to be able to evaluate whether the information circulates correctly between them. In some specific cases, let to the evaluation leader's appreciation, interviews can be made rather than a questionnaire mailing.

• *The answers analysis*
The answers are then analysed with the help of our guide. The first action consists in considering whether all the roles of the maintenance are clearly identified. For this purpose the information flow of the project is drawn with the given answers of the first five questions. The information flow is then compared to the "correct" (in theory) information flow as shown figure 2. In this theoritical model, all the roles played in a maintenance process are exhibited. Several roles can be played by a single actor but the roles must be clearly identified. Comparing the model of the project and the theoritical model allows to exhibit missing role identification, and then correct communication problems.

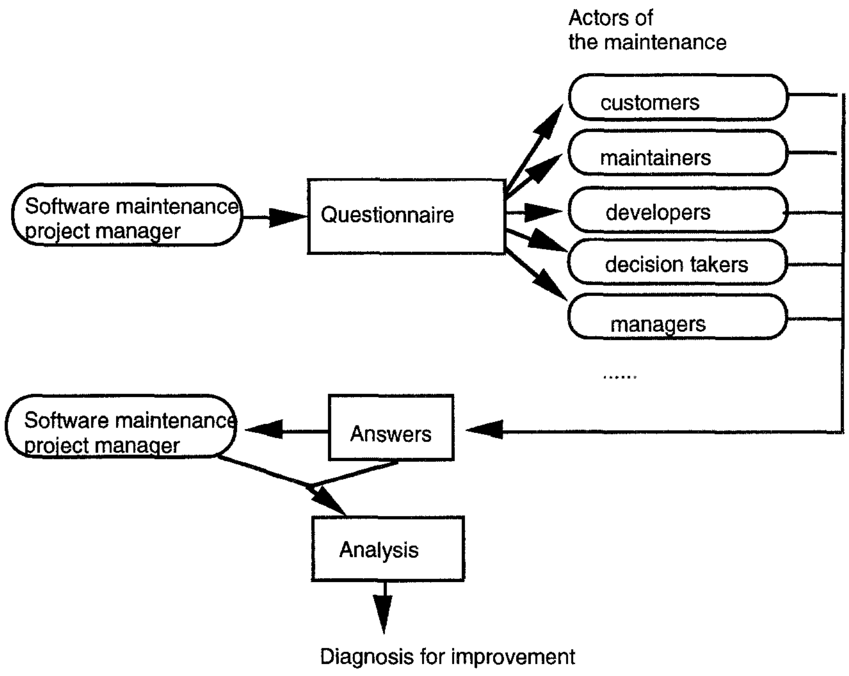
Actors of the maintenance
customers
maintainers
developers
decision takers
managers
......
Software maintenance project manager
Questionnaire
Software maintenance project manager
Answers
Analysis
Diagnosis for improvement

Fig. 1. The process set up in the approach

This first step allows to display a taxonomy of all the maintenance actors. Every actor can be put in a certain class corresponding to a role, and then answers can be gathered by roles, to facilitate their evaluation.

The second step of the analysis is made with an evaluation grid (see example on appendix B), which allows to sort the answers. At the end of this step, the main tendancy of the problems encountered is given and a diagnosis and solutions can be considered.

• *Diagnosis and solutions for improvement*

For each class of problems (communication, techniques, strategy) general solutions for improvement are given. Roughly, three types of solutions are given:
- If a communication problem has been detected, an organisation identifying clearly each one's role must be set up formally. It is important to point up at this step that all the roles identified in figure 2 must be clearly fulfilled by a person. Even though an actor may have several roles, this person must realise that he/she plays several roles and must behave in accordance with his/her role.
- For technical problems, tools must be used, documentation written, and the organisation formally described, by following software engineering rules.
- If the analysis showed up that the software is not strategic for the company, efforts must be put on organisation and in finding replacement solutions. Moreover, it is devoted to the maintenance manager in that case, to find ways of motivation for the maintenance team. Motivation can obviously come from a deep involvement but also

by a new expertise acquisition. Choosing to use a new tool at this stage may be an efficient mean of remotivating people.

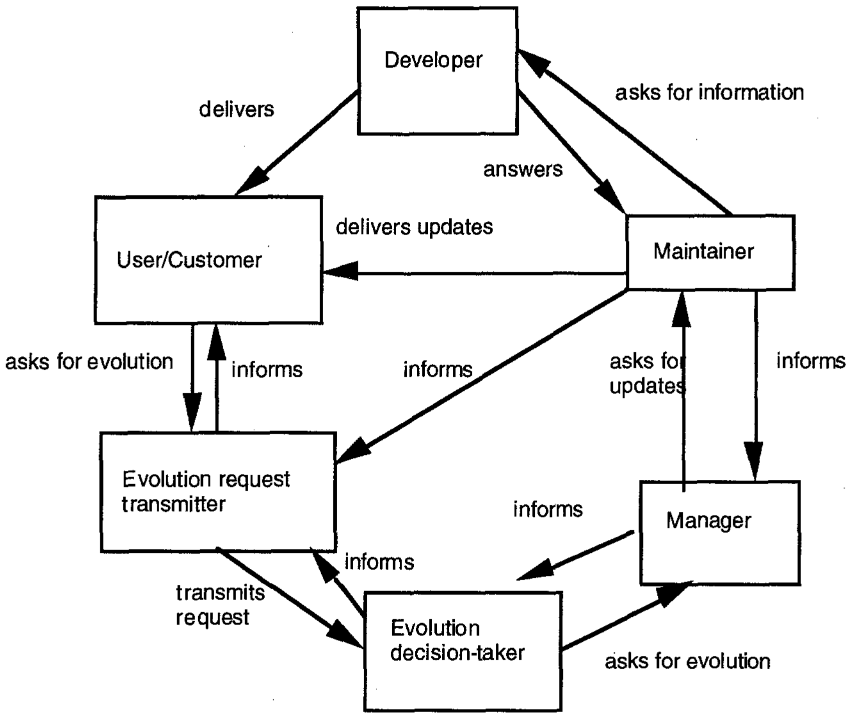

Fig.2. Information flow between maintenance roles

Application on a project

This guide has been experimented on a project currently being maintained by a team of 7, for 15 years and which will not be replaced by the year 2000.

As it was the first experiment, the evaluation has been carried out by the maintenance group of the QSE team and not by the maintenance project manager.

We asked the manager a list of the people involved in the maintenance process. He provided us with a list of 25, composed of developers, customers (including evolution request transmitter and evolution decision-taker) and maintainers. We first sent the questionnaires but we also decided to go interview the maintainers, who wished to have opportunity to talk about their job. On a total amount of 25 questionnaires, we collected 15 answers.

The first observation that was made concerned the communication within the project. The evaluation pointed out that some roles were not clearly identified. It did not mean that the first questions were answered by "I don't know", but it emphasized differences between the answers. In fact, decision-taking roles were sometimes given to different people and this situation led to conflicts. A first way of improvements consisted then to formally describe the organisation to put in place. Once described, the organisation had to be supported by making everyone aware of roles and people within the project. The main success factor for this first improvement is a strong involvement of hierarchical management, who must formally define everyone's task. The second observation was that since the project was strategic for the company, efforts and resources had to be put on it. Our diagnosis allowed the managers to clearly realise this situation and led to a deliberate decision on maintenance improvements, beginning by resources allocation.

Finaly, some technical remaks allowed to enhance the maintenance process, by improving the configuration management. Technical points seemed to be the less serious defects to improve in the organisation we evaluated. We nevertheless realised that technical novelties were necessary for the maintainers to feel comfortable, as they tend to find technical solutions in term of tools. Taking their specific problems into account imposed to make technical answers.

4.Conclusion

A first use on a project was of a great help for us, as it allowed us to realise whether our guide could be utilised as such or not. It showed that a quick diagnosis for managers who are too involved in their project to see clearly the problems is very useful. Nevertheless, some improvements on the questionnaire are necessary in order to have a more accurate diagnosis at the end. This is the second phase of our work , which will lead to an improved guide that we plan to send to other projects in maintenance phase.

The first draft of the document clearly proved how important it is for maintainers to be considered and assisted from the outside. Despite our first fear, we were warmly welcomed within the projects, that needed and asked for help, and the first results we obtained, although looking obvious to us, seemed almost to be a "revelation" to the manager. This reaction proves, if necessary, that software maintenance in a scientific domain, is more than ever a subject to work on, with a need of concrete help to provide to projects.

Appendix A : the questionnaire

Date: .. / .. / .. **Maintenance of the project:**
Name Role in the project
C1: Who did develop the code?
C2: Who does use the code?
C3: Who does ask for evolution?
C4: Who does make it evolve?
C5: Who does decide for evolution?
T1: Where is the code located?
T2: Where is the documentation located?
T3: How is the project history managed?
T4: How are the correction requests managed?
T5: Does it exist a culture (programming rules...) within the project?
S1: Is the code strategic for the company?
S2: What is the life expectancy of the code?
S3: Where is the "know-how" in the code?
S4: Who must control this "know-how"?
S5: Is there any marketing plan for this code?

Appendix B: Examples of evaluation grid

General observation on the answers

Evaluation criterion	Is there a general agreement on the answers?
Observation	Yes
Possible interpretations	A general agreement may mean that the maintenance organisation is satisfactory with a good information flow It may also mean that the project is small enough for th eproject to be managed only by one or two people with no problems. A common and strong culture within the project can also lead to identical answers. Finally, giving the questionnaire to too few people with the same kind of roles may lead to an identity in the answers.
Observation	No
Possible interpretations	There are answers but they are different or there is sometimes no answers. Sometimes, an absence of answer may be admitted. In such case, see the evaluation grid for each specific question. Differences in the answers may be a problem because it could come from "incorrect" knowledge of some of the actors. It could also point out a lack of communication between the different classes of actors.

Particular observation on question T4 asked to a user/customer

Evaluation criterion	Is there an answer?
Observation	No
Possible interpretations	There is no formal process for problem report, evolution request,...
Observation	Yes
Possible interpretations	Answers should be compared to answers from other classes of actors in order to see if the process is the same for everyone.