

Update on Software Maintenance

With the migration to Y2K and to Euro, organizations have experienced the 2 largest maintenance projects in Software. Yet, Software Maintenance has a modest place in specialized journals and in professional seminars.

Should one infer that software maintenance is not a major concern for IT managers any longer? Operational reality contradicts this apparent lack of interest. Indeed, mastering software maintenance is becoming critical. There are four reasons to that :

- Maintenance costs which already are very high, shall continue to increase and become the more visible as the prices of hardware decrease,
- A fast response of organizations to adapt themselves to the business environment is a key success-factor. This flexibility depends on their capabilities to quickly adapt their strategic applications, i.e. on how well they can control *evolutive* software maintenance.
- Organizations should also master *adaptative* software maintenance if they wish to make available their existing applications (such as order processing, delivery tracking...) to their internet-customers.
- While giving access to their applications from the Internet, organizations expose their level of quality, and therefore must master their IT processes, notably regarding *corrective* maintenance.

One should stress that the issue of software maintenance is not exclusively related to old applications which could be replaced by new ones in order to solve the problem. In fact, as soon as a new application is in use, it get into the maintenance process.

Facts and Figures

Two studies illustrate the problem of maintenance. The first [1] one identifies the three most negative factors impacting software maintenance :

1. the absence of process (in 73% of the cases),
2. the lack of up-to-date documentation (in 68% of the cases)
3. the lack of time to update the documentation (in 54% of the cases)

A second study [2] shows that the costs of maintenance increase as time passes since the many modifications brought to the source codes complexify the interventions.

Solutions

In order to improve the quality of the codes and to reduce delays and costs, the three most efficient approaches are :

1. setting up a maintenance process,
2. documenting or using an automatic redocumentation of software applications
3. renovating applications.

Before discussing in details these solutions, it is important to stress that their successes do not depend only on their intrinsic qualities but also in their acceptance by the developers. Developers indeed are in fact craftsmen, in the best sense of the word, who usually are adverse to imposed solutions. The successes of the approaches which are implemented therefore rest in great part on listening to the needs of the field, and on the solutions solving the daily problems (bottom-up approach).

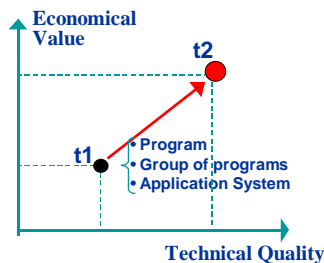
The Software Maintenance Process

To implement a software maintenance process requires that two key activities be fully mastered regarding Software Engineering :

1. Software Configurations Management.
2. Change management.

These two well-known activities shall not be detailed here. However, it is important to stress another activity which is relatively little used : *measurement*. Indeed it is not possible to control a process without metrics. Apart from elementary indicators such as days*man-day, or number of Lines of Code modified per person, it is important to measure :

- the technical quality of each component,
- the economical value of each component,
- the evolution of metrics over time.



This should be weighted with the context of the application, and it should be possible to aggregate these measures *vertically* per groups of programs and per application, and *horizontally* according to features such as portability, reliability, ease of maintenance...

Once the domains of non-quality are identified, a decision model should be available in order to choose the corrective actions.

Software Redocumentation

Documentation has always been the weak link in the set of deliverables related to software projects. This is confirmed by a study by IEEE which shows that 95% of software documentation is obsolete and or not complete. The study [1] shows that in 89% of the cases, developers are obliged to use the source code as the main source for documentation. It follows that the greater part of the maintenance time is spent looking for information, reading it and trying to understand it rather than modifying the code.

A really useful documentation is more than just comment lines explaining a line of code or a group of instructions. It includes all the information required for the maintenance of the application, notably :

- how the application is organized (who is the owner, what is the functional or structural decomposition),
- which components are involved (JCL, programs, files and data bases, transactions, etc...),
- what are the relations within a component and between components,
- the Data Dictionary.

The wealth of information is not the only criteria of a good documentation. This documentation should also be up to date, available to all, and easy to use, which implies :

- processes for automatic update and shared access,
- adaptativeness to the user's cognitive model,
- easy navigation within a component and between components,
- a familiar usage metaphor,
- a dynamic capacity for impact analysis intra- and inter-components

Since application knowledge resides in the source codes and with the experts, a good redocumentation system should include an automate for analyzing the sources, and an infrastructure to capture the experts' knowledge.

Today, it is possible with the technology to find and automatically exploit all the knowledge contained in the source codes. In addition to this knowledge recovery advantage, one should add that the source codes are the most valuable source since they contain the most up to date information.

As an example, the automated re-documentation [3] of an application in the insurance industry comprising about 5 million LoC, generates around 500 000 HTML pages of exhaustive documentation. This plainly shows that :

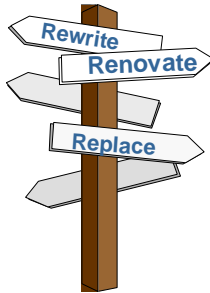
- without an automated process, such a large volume of documents cannot be either generated nor kept updated within a reasonable budget and period of time.
- without hypertext navigation, such an amount of information cannot be exploited.

It is not possible to completely automate the capture of the experts' knowledge, called *semantic assignment*. However it is facilitated on one hand by a man-machine interface consisting of a documentation tool with data-entry fields, and on the other hand by the technology of object-data bases which allows an automatic storage and management of this knowledge. Indeed, a comment is only one of the attributes of an object in data-processing.

Renovation

The first goal of renovation is to improve the maintainability of code in order to reduce maintenance costs. Beyond this, renovation has an impact on the strategy for the evolution of applications. When an application becomes too complex to maintain, management faces two options :

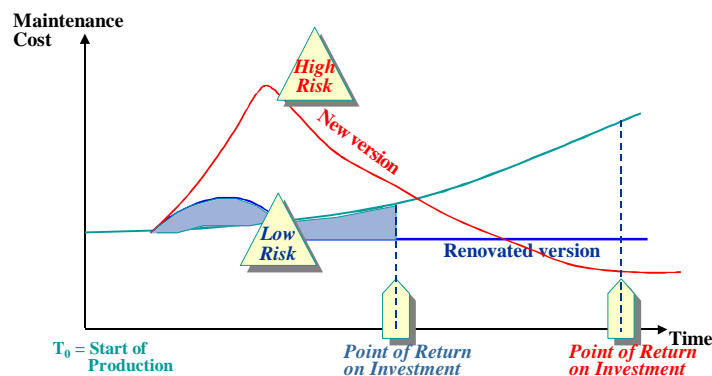
- re-write the application,
- replace the application by another or by a software package.



In some cases these options cannot be avoided, but are high-risk, because they are of the « big-bang » type, and are costly.

The third way , Renovation, which is now possible thanks to current technologies, leads to a quick reduction of maintenance costs, to an extension of the lifetime of the application, and therefore to the decrease of the global cost. Furthermore, renovation is a low-risk solution because it allows for reverse actions, can be divided in activities of reasonable size, and can be automated in great part.

[Economics of SW Re-engineering, Harry M. Sneed](#)



Renovation covers Redocumentation, Restructuration, Restauration and Reengineering techniques. Redocumentation was discussed hereabove. Restructuration includes a number of techniques which are very automated such as :

- the removal of dead codes and of unused variables,
- the reduction of the number of GOTO,
- the removal of backward GOTO,
- the restructuration of data,
- the deletion of unnecessary files,
- the reduction of « Dominance Trees » (for example, reduction of nested IF),
- the alignment of the precision of numerical data,
- the standardization of code and of names.

In a later stage, it is possible to tackle operations such as :

- restoring the meaning of functions
- re-engineering

These operations require more human interventions, but are made easier with the tools described hereabove, and by other automated techniques such as « slicing » which allows the grouping of the Lines of Code according to the type of processes (display, access to data, etc...) and to assign them to functions (concept of « Function Mining »).

A Renovation Project should start with an initial diagnosis. It rests on measuring the quality of the source code and of the economical value of the components. It allows to choose : the sub-set of components with the highest economical value in the application, the most adequate renovation techniques for this subset.

The technical and economical metrics of components and of applications which have been previously mentioned play a key role in the Renovation process. Not only do metrics allow targeting the renovation actions , but it allows monitoring how quality evolves over a period of time, and therefore to correct any deviation through specific actions.

Technologies and Architectures

Redocumentation and Renovation rest on various technologies :

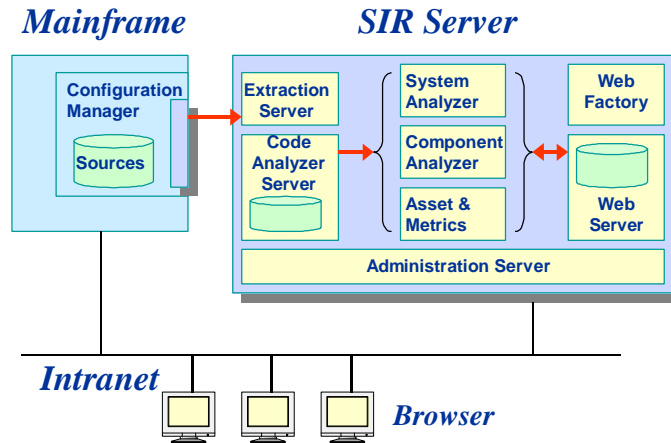
- Analysis of Source Code.
- Manipulation of Source Code.
- Information storage and diffusion.

For analysis and transformations, the tools which operate directly on the Source Code have generally limited capabilities. The most advanced tools use techniques which proceed from work done on compilers. Indeed, a compiler is the only automate which can interpret correctly and exhaustively the grammar and the syntax of a source code. To that end, it builds up an abstract representation of the source code in the form of a tree structure adapted to the construction of control flows and of data flows, and to the propagation of modifications.

Storing in form of objects is in conformity to the nature of programmatic objects which are very interdependent. It allows the enrichment of information and to interrogate it with a formal language.

Regarding diffusion and visualization of information, the Internet technology - based on centralized servers for the creation of HTML and XML pages - and workstation-servers provided with navigators is very adapted to the documentation of applications. It offers major advantages :

- The concept of a central server allows to control the integrity of documentation, its synchronization with of software configurations, and the accesses,
- The navigators alleviate the installation and the maintenance of the proprietary code on each workstation,
- Intuitive navigation thanks to hypertext and hypergraph interactions, while XML makes possible to exploit the full content of the collected information,
- The Web metaphor reduces the training time to a few hours at the most.



An architecture based on client/server models allows the creation of « documentation servers » which have no impact on the mainframes nor on the workstations. Moreover, the servers can be spread over several computers in order to reach the desired scalability.

It is possible to ask oneself why the solutions which are presented here are not more widely used. There may be several reasons to that :

- Very few companies master the compiler technology,
- Furthermore it is also necessary to master the mass production process required for the treatment of application systems of several millions LoC,
- Few people have the know-how to productize such technologies,
- Until the relatively recent past, the required processing capacities and memory sizes were an obstacle to solutions which were economically acceptable,
- The Internet technology which forms the basis for the diffusion of information is also a relatively recent tool for maintenance teams,
- The traditional suppliers of redocumentation software tools are handicapped by the old concepts: workstations oriented (instead of servers), proprietary presentation interfaces, low performance of the data base implementation when millions of lines of codes are to be .

Surprisingly, it is the software houses which came last into the software maintenance market who have been able to capitalize on all these new technologies and which offer well architected, evolutive, and economical solutions.

Benefits which can be measured

The solutions which have been recommended in this paper have been implemented in real cases that have provided valuable information :

1. Compared to a traditional documentation approach, using an *automatic hypertext* redocumentation and navigation tool [3] has allowed to reduce research time from 5 hours to 1h30.
2. In another project [4], the improvement of the maintenance and of the documentation processes, and the restructuring of the application codes have allowed to process 9% more of requests in 18% less of man-days.

From the qualitative point of view, one should stress the need to provide maintenance teams with efficient tools having a high technological content in order to boost their motivations.

As for management, using these tools allows them to regain control of the maintenance process and put them in a position to engage into a real quality policy.

CONCLUSION

It seems obvious that the solutions presented here have a promising future as they help addressing many challenges:

- The human resources shortage, most software development engineers being engaged in software maintenance tasks,
- The continuously growing number of new applications systems, and their increasing complexities,
- The variety of computer languages and of components used
- The pressure on costs and quality.

These solutions provide a significant leverage effect on software quality and on IS budgets.

They are the first tools for the industrialization of the software asset management, and shall have a decisive impact on the competition between companies at the start of the 21st century.

Note : This article has been written by Patrick TOCA, Co-Managing Director of NGSET, 30 Avenue Edouard Belin, 92500 Rueil Malmaison, France. All rights reserved. Copyright by NGSET.

- [1] M.J. Castro Sousa and H. Mendes Moreira, *A Survey on the Software Maintenance Process*, O-8186- 8779-7/98 IEE
- [2] HM Sneed, *Economics of Software Re-Engineering*, *Journal of Software Maintenance, Research and Practice*, Vol 3, N°3, Sept 1991
- [3] SIR software solution by NGSET.
- [4] G. Visaggio, *Process Improvement through Data Re-Use*, *IEEE Software magazine*, July 1994.