What you Need to Know about Software Maintenance

Article · January 2005

CITATIONS
4

READS
5,594

3 authors, including:

Alain April
École de Technologie Supérieure
157 PUBLICATIONS **1,420** CITATIONS
SEE PROFILE

Alain Abran
École de Technologie Supérieure
654 PUBLICATIONS **9,556** CITATIONS
SEE PROFILE

WHAT DO YOU NEED TO KNOW ABOUT SOFTWARE MAINTENANCE

Alain April, A. Abran and R. Dumke

Software accounts now for a increasing share of the content of modern equipments and tools, and must similarly be maintained to ensure its continuous operational efficiency. Although the maintenance of the equipments is discussed extensively, very little is published about software maintenance and how it affects us. This paper presents an overview of key topics of software engineering maintenance.

Has your production ever been stopped because of a software problem?

Software maintenance is indeed required to support many key equipment and product lines throughout their daily operational cycles. For instance, software-related problems and modification requests are sent to the supplier of the product where it is logged and tracked, the impact of proposed changes is determined, software code is modified, testing is conducted, and a new version of the software product is released. This looks quite simple, but if it were that simple, the software fix would be applied in minutes. Then why does it often take days, weeks and sometimes months to dot it? Because software is sometimes quite complex and that even a seemingly very minor change to an element might have a very sextensive impact throughout the whole structure, should such element be used across the structure of either the software itself or throughout the operation system linked to such software.

Software maintenance is often perceived merely as fixing bugs, that is reactive to errors and omissions. However, studies and surveys over the years have indicated that there is much more to software maintenance than merely fixing bug. Some studies have even reported that a corrective software maintenance represents less than 20% of the maintenance workload. . The consensus on the key components of the software maintenance process has been documented in the ISO/IEC 14764 the International Standard for Software Maintenance. This ISO this standard, although as well known as ISO900:2000, is important to the software maintainers and to general management for understanding better the services provided on the software you own and that you are about to service or modify.

	Correction	Enhancement
Proactive	Preventive	Perfective
Reactive	Corrective	Adaptive

Figure 1: ISO14764 software maintenance categories

The ISO/IEC standard recognizes four categories of maintenance work(see figure 1) :

- Corrective maintenance: Reactive modification to a software product, performed after delivery to correct the problems identified.
- Adaptive maintenance: Modification to a software product, performed after delivery to keep a software product usable in a changed or changing environment.
- Perfective maintenance: Modification to a software product, after delivery to improve performance or maintainability.

- Preventive maintenance: Modification to a software product, after delivery to detect and correct latent faults in the software product before they become effective faults.

In well-managed software organizations, most of the changes to software are carried out to adapt such software to the changing business or technology environment. Because the business context now moves very quickly the equipments must also improve constantly, new and better functionality need to be inserted in the existing software. The software does not deteriorate physically with time and does not age when it operates. It. However, due to continuous additions or modifications, it gets more complex and patched with the numerous changes and progressively becomes more difficult to maintain.

These are some of the issues that have to be recognized and understood for maintaining management control over the maintenance budget of both the software and of the related equipment. So next time you look into the list of software problem try to separate them and identify the real % of corrections. You'll see that it does not account to so much. It's the changes that take the most out of your budgets. And changes , by definition, have more to do with maintenance projects than routine maintenance. It becomes evident that some of those requests will take longer because they are not routine work!

But why does it take so long to change the software?

A number of very complex issues must be dealt with to ensure adequate maintenance of software systems. For example, it is most challenging for a software maintainers to analyzed a 500,000 or 2,000,000 lines of code software system that the maintainer did not develop himself to find a hidden defect or to identify where a specific change must be implemented. Furthermore, software engineering is far from a mature engineering disciplined and unlike mechanical engineering, still too little is provided in terms of professional and accredited training to the software maintainers. Instead it is often observed that software maintainers have a limited understanding of the products they must maintain. Over the years, both practitioners and researchers have reported that up to 60% time spent on maintenance is indeed devoted to developing a good understanding the software to be modified, prior to initiating any change to it. This of course leaves much less time to carry out the change and to test it extensively

If it's not the maintainer's fault whose is it?

Why is software so hard to maintain? Many of such difficulties can be traced back to the software development process itself which often does not take into account the maintainability requirements: too often software is developed in uncontrolled environments (e.g. read 'hackers style') and not to professional engineers standards. In industry, if a product or an equipment is not built to the proper quality standard, and without adequate maintenance documentation, then maintenance will be abnormally high when compared to products or equipments developed to the highest maintenance standards.

Because software is often embedded (that is, hidden) into industrial products it lacks visibility, suffers from lack of management attention, then from lack of resources, which often leads to lower quality: for instance, when a trade-off must be made in a situation of schedule compression, then software is often where development cuts happen rather than on the more visible hardware related components. Of course, software vendors are part of the problem; in making you believe that their software is better, faster and …maintainable (but with little supporting evidence).

The software maintainability issue is often quoted in the Information Technology industry. The IEEE Computer Society [IEEE610.12] defines "maintainability" as the ease with which software can be maintained, enhanced, adapted, or corrected to satisfy specified requirements. ISO/IEC defines maintainability as one of the main quality characteristics of a software [ISO9126].

Maintainability of software must be specified, reviewed and controlled during the software development activities if we wish to ever properly manage the maintenance process and subsequently reduce the maintenance costs. If this is done successfully, the quality of maintenance of the software (its maintainability) will likely improve.

Unfortunately, there is often a lack of attention to maintainability during the software development process. Often software disregards this key business and engineering requirement. Time and time again a software is implemented and send to the operations without adequate maintenance documentation, unduly adding later on considerable maintenance cost wherever a software change must be implemented.

Aren't there best practices for improving software maintenance?

The need and benefits of mature engineering processes is well documented, including for software development. Similarly, there is a well recognize link the levels of process maturity and related costs savings in software maintenance. For example, the Software Maintenance Capability Maturity model (SM^{CMM}) identifies the best practices associated with the software processes unique to a maintainer. SM^{CMM} was designed as a customer-focused benchmark for either:

- Auditing the software maintenance capability of a software maintenance service supplier or outsourcer; or
- Internal software maintenance organizations.

Some of the activities unique to software maintainers are:

- Transition: Is a controlled and coordinated sequence of activities during which a system is transferred progressively from the developer to the maintainer;
- Service Level Agreement (SLA's) & specialized maintenance contracts: Maintainers negotiate SLA's and domain specific contracts;
- Help Desk handling of MR's and PR's: Maintainers use a problem handling process to prioritize, document and route the requests they receive;
- Acceptance/rejection of MR's: Maintainers will not accept modification requests work over a certain size/effort/complexity and will reroute these requests to a developer;
- Impact Analysis (refer to II.1 impact analysis);
- Regression Testing: Maintainers need to "perform regression tests on the software so that the new changes do not introduce errors into the parts of the software that were not altered".

The SM^{CMM} has been developed from a customer perspective.. The ultimate objective of software maintenance improvement programs initiated as a result of a SM^{CMM} assessment is increased customer (and shareholder) satisfaction, rather than rigid conformance to such a model.

A higher capability level, in the SM^{CMM} context, means, for customer organizations:

a) Reaching the target service levels and delivering on customer priorities;
b) Implementation of the best practices available to software maintainers;
c) Obtain transparent software maintenance services and at costs that are competitive;
d) The shortest possible software maintenance service lead times.

For a software maintenance organization, achieving a higher capability can result in:

a) Lower maintenance and support costs;

b) Shorter cycle time and intervals;

c) Increased ability to achieve service levels; and

d) Increasing ability to meet quantifiable quality objectives at all stages of the maintenance process and services.

In the SM^{CMM} model, the key software maintenance processes have been grouped into three classes (Figure

2).

a) Primary processes (operational);

b) Support processes (supporting the primary processes); and

c) Organizational processes offered by the IT unit or by other departments of the organization (for example: finance, human resources, purchasing, etc.).

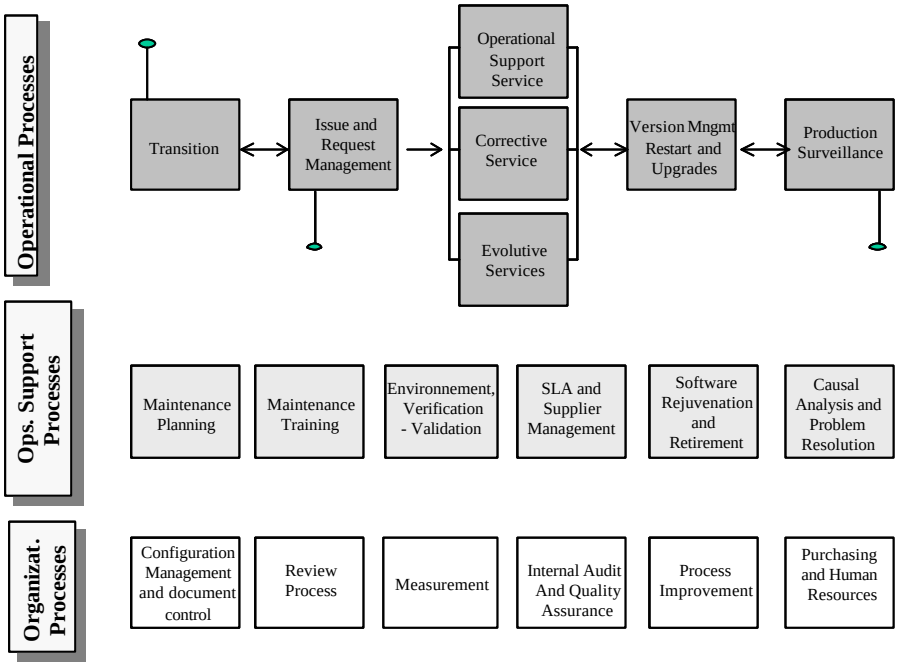

Figure 2: A classification of the Software Maintainer's Key Processes

The key operational processes (also called primary processes) that a software maintenance organization uses must be initiated at the start of software project development and then maintained subsequently, beginning with the transition process. The *Transition process* ensures that the software project is controlled and that a structured and coordinated approach is used to transfer the software to the maintainer. In this process, the maintainer will focus on the maintainability of this new software.

Once the software has become the responsibility of the maintainer, the *Issue and Service Request Management process* handles all the daily issues, problem reports, change requests and support requests. These are the daily services that must be managed efficiently. The first step in this process is to assess whether a request is to be addressed, re-routed or rejected (on the basis of the service-level agreement and the nature of the request and its size). Accepted requests are documented, prioritised, assigned and processed in one of the service categories: 1) *Operational Support process* (*which typically does not necessitate any modification of software*); 2) *Software Correction process*; or 3) *Software Evolution process*.

It is to be noted that a number of service requests do not lead to any modification to the software. In the SM^{CMM} model, they are referred to as 'operational support' activities, and these consist of: a) answering to questions; b) providing information and counselling; and c) helping customers to better understand the software, a transaction or its documentation.

The last two main operational processes are the *Version Management process*, to move items to production in a controlled fashion, and the *Production Surveillance process*, which will ensure that the operational environment has not been degraded. Maintainers must also monitor the behaviour of the operational system and its environments for signs of degradation. They will quickly warn other support groups when something unusual happens (operators, technical support, scheduling, networks and desktop support) and judge whether or not it is an instance of service degradation which needs to be investigated.

A process which is used, when required, by an operational process is said to be an operational support process. In this classification, we include: a) the many maintenance planning processes; b) the maintainer's education and training; c) the maintenance environments and testing; d) management of the contractual aspects and service level agreements; e) rejuvenation or retirement of software; and, finally, f) resolution of problems. These are all key activities, which are available to support some operational process activities.

Organizational processes are typically offered by the IT department and by other departments in the organization (for example: human resources, finance, quality assurance and ISO9001). While they are important to measure and assess, it is often easier for the maintainer to start defining the operational and operational support processes.

This generic model should help understand and position the various key software maintenance processes. What is important is that these processes be explicitly listed and classified based on their type (operational, support or organizational). The **SM**$^{\mathbf{CMM}}$ was developed in an industrial environment with practices recognised as useful).

In summary software maintenance is not all that simple and a maturity model might be helpful at assessing your suppliers maintenance maturity. Its proven …in engineering, better maintenance leads to lower operational costs.