Otto-von-Guericke Unversität Magdeburg

SM^{mm} Model to Evaluate and Improve the Quality of Software Maintenance Process

Alain A. April

Institut für Verteilte Systeme
Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg
Postfach 4120
39016 Magdeburg
Germany

Preprint Nr. XX
2004

SM^{mm} Model to Evaluate and Improve the Quality of Software Maintenance Process

Arbeitsgruppe Softwaretechnik – Software Measurement Laboratory SMlab
GÉLOG – Laboratoire de Génie Logiciel

Alain April

Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik - Institut für Verteilte Systeme
Postfach 4120, D-39016 Magdeburg

http://ivs.cs.uni-magdeburg.de/sw-eng

ABSTRACT

The software maintenance function suffers from a scarcity of management models that would facilitate its evaluation, management and continuous improvement. This paper presents a maintenance-specific capability maturity model: Software Maintenance Maturity Model (***SM***^{mm}). This model adopts a similar structure and should be used as a complement to the CMMi©[1] (Capability Maturity Model Integration of the Software Engineering Institute) developed by Carnegie Mellon University. This ***SM***^{mm} is based on the seminal literature on software maintenance, international stndards and practitioners' experience.

[1] CMM and CMMi is a trademark of the SEI of the USA.

Thanks

We thank industry members who have worked on this project over the years, including special thanks to Mr. Dhiya Al-Shurougi for his most valuable field study conducted in the Middle East.

Remarks

This research project is being carried out at the Software Engineering Research Laboratory at the École de Technologie Supérieure–University of Québec, headed by Dr. Alain Abran.

Content of the study

Table of figures

Abbreviations and acronyms

CobiT	Describes the objectives, generally accepted, of information technology control for management and internal auditors [*Cob00*]
CMM©	Capability Maturity Model (version 1.1) published by the SEI [*Sei93*]
CMMi©	Recent integrating version of the CMM published by the SEI [*Sei02*]
IEEE	Refers to the Institute of Electrical and Electronic Engineer
ISO	Refers to the international standard organization
ISO JTC1/SC7	Sub-commity, which develops international standard for software
ISO90003:2004	Interpretation guide of ISO9001 for the software industry [*Iso04*]
ISO9001:2000	Current standard ISO9001 published during year 2000 [*Iso00*]
ISO/IEC12207	International standard for software life-cycle [*Iso95*]
ISO/IEC14764	International standard for software maintenance [*Iso98*]
ISO/IEC15504	International standard concerning software improvement models and methods. There is nine documents [*Iso98a*]
MR	Maintenance Request (or report) for operational software
PR	Problem Request (or report), which takes priority over current maintenance work needed to repair operational software rapidly.
SEI	Software Engineering Institute of Carnegie Mellon University in U
SPICE	Name given to the ISO/IEC 15504 [*Iso98a*] during its inception
Xtreme	Development methodology issue from the Internet industry [*Jef00*]

1 RESEARCH MOTIVATION

In the new global competitive context, organizations undergo pressures from their customers. Customers are becoming more and more demanding and ask for high quality services within the shortest schedule, at the lowest possible cost and followed by post-delivery services that beats the competition. To satisfy the quality, quantity and the ever-challenging service levels, the dynamic organizations must have access to a software portfolio to supports their business processes. This software portfolio must be reliable and therefore very well maintained.

Maintaining the mission critical software portfolio of an organization is not an easy task and requires a management system for software maintenance. To be adequate a management system for software maintenance has to satisfy the service criteria of its customers and the technical criteria of the domain to maximize strategic impact and to optimize the economical criteria of the activities of the maintenance of software.

To achieve these many concurrent objectives, continuous improvement of the software maintenance function must be a priority in order to pursue progressively these objectives. Such continuous improvement process must also allow for a progressive approach that is adaptable to every software maintenance organization. However, there is currently a lack of specific process improvement models for the software maintenance function. The software maintenance organizations do not currently have access to such improvement models to determine the best improvement strategy.

This lack of maintenance-specific capability maturity models is the research problem that was chosen for this article.

As early as 1987, Colter [*Col87*] highlighted that "the greatest problem of software maintenance is not technical but managerial". If the management problems are listed as the key problems of software maintenance, then the technical aspects are not far behind. Numerous publications address the problems associated with resources, processes and toolsets of software maintenance. These documented problems vary according to the specific perspectives taken by the authors.

This article presents a maintenance-specific capability maturity model: The Software Maintenance Maturity Model – (***SM^{mm}***).

Section 2 and 3 presents the findings and contributions from the software maintenance and capability maturity models literature review including a discussion of what is missing in the CMMi© to reflect the maintainer's unique processes and activities. Section 4 identifies the research questions. Section 5 presents the research methodology. Section 6 introduces the architecture and design process of the proposed ***SM^{mm}***. It also describes the approach taken to build the model, as well as the model purpose, scope and foundation. This is followed in section 7 by examples of the content of two key process areas: Maintenance performance management process and Management of Service Requests and Events. Finally, the research contributions are presented in section 8 followed in section 9 by an overview of further work in progress.

2 SOFTWARE MAINTENANCE LITERATURE REVIEW

2.1 Introduction

The software life cycle can be divided into two distinct parts [*Mor96, s19.1*]: a) the initial development of software; and b) the maintenance and operation of the software. Included, in table I, an example of the many definitions of software maintenance that can be found in literature:

Table I

Generally accepted definitions of software maintenance

"Changes that are done to a software after its delivery to the user"	Martin & McLure [*Mar83*]	1983
"Maintenance covers the software life-cycle starting from its implementation until its retirement"	Von Mayrhauser [*Ben00*]	1990
"Modification to code and associated documentation due to a problem or the need for improvement. The objective is to modify the existing software product while preserving its integrity."	ISO/IEC 12207 [*Iso95*]	1995
"The modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment."	IEEE 1219 [*Iee98*]	1998

Lehman [*Leh80*] states that "change being unavoidable forces operational software to evolve or they progressively become less useful and ultimately obsolete". Maintenance becomes unavoidable for operational software used daily everywhere in the company. This point of view emphasizes that the software maintenance definitions aim primarily at application software (as opposed to base software like operating systems).

2.2 Difference between Operations, Development and Maintenance

The processes and activities of the computer operations domain are distinct from the processes and activities of the software maintenance domain [*Iso98, s1*]. It is specified in the ISO/IEC14764 international standard that operation activities like: a) backup; b) recovery; c) operating system administration and; d) computer operations are carried out by the computer operation personnel of the data center. These activities are not part of the software maintenance scope of work as presented by the many definitions of table I. Although this is well stated in the ISO/IEC14764 international standard it is common that managers confuse the computer operations and the software maintenance processes and activities. This confusion might stem from the fact that both organizations often work closely to one another. There is an important and very active interface between software maintenance and computer operations that aim especially to assure that the infrastructures, that support the operational softwares are operational and responsive (change management, service calls concerning a failure in production, recovery of the environment and data after a disaster, recovery of data, automated scheduling, disk management and tape management) [*Iti01a, Iti01b*].

On the other hand, how clear is the difference between software maintenance and software development to managers? Software development can also be confused with software maintenance [*Abr01 s3.2.2*]. The difference can be more difficult to explain to management when the developer of the software also carries out its maintenance. This confusion originates, principally, from the fact that some processes and activities of software maintenance are similar to the ones of software development (analysis, design, coding, configuration management, testing, reviews and technical documentation). In practice these similar activities differ because they are adapted to the specific context of maintenance ISO/IEC 12207 [*Iso95 s5.5.3*]. A key difference is that the maintenance work is carried out by only one or two maintenance staff and for very short-term deliverables. It is true to say that a software maintenance employee can obtain part of his expertise and knowledge from the same teaching sources and training than that of his colleagues of the software development.

It is to note, equally, that the organizational structure of the software development teams is principally in the form of a structured project using project management techniques. The development project typically is created for a temporary and fixed length and is not sustained after the delivery of the software. The software development project team develops a plan of resources, profits and loss, specific deliverables and objectives and aims a planned date for closing the project. The structure of the maintenance team is very different for it must face the events and daily requests of the customers while maintaining the continuous service of the operational systems under their responsibility.

In a software maintenance organization, it is important to understand how the management of maintenance processes and activities differs from the management of software project processes and activities. While project management is organized towards the delivery of a product within a specific timeframe and by a pre-arranged project closure date, the maintenance organization and processes must be structured to handle ongoing work on a daily basis for its customers with, by definition, no closure date. Key characteristics of the nature and handling of small maintenance requests have been highlighted in [*Abr93*], for example:

1. Modification requests come in more or less randomly and cannot be accounted for individually in the annual budget planning process;
2. Modification requests are reviewed and assigned priorities, often at the operational level – most do not require senior management involvement;
3. The maintenance workload is not managed using project management techniques, but rather queue management techniques;
4. The size and complexity of each small maintenance request are such that it can usually be handled by one or two maintenance resources;
5. The maintenance workload is user-services-oriented and application-responsibility–oriented.
6. Priorities can be shifted around at any time, and requests for corrections of application errors can take priority over other work in progress.

When a user submits a modification request (MR) it is necessary to estimate the effort needed to modify the existing software. The study of Dorman & Thayer [*Dor97*] states that modification requests (MR's) and problem reports (PR's) go though an investigation and impact analysis activity, which is unique to software maintainers. If the estimated effort is too big on a modification request it will be sent to a software development team and treated as a project.

There is, for software maintenance, a unique process that accepts/rejects work demanded in a modification request. This process takes into account the size and effort of a specific modification. April [*Apr01*] presents the process used at Cable & Wireless where the maximum effort of an MR that a maintenance programmer will accept is five days of effort. It is very different for Problem Reports (PR's) where, whatever the size or effort required to fix a failure, it will be processed immediately by the maintenance staff. This five day limit is also recognized by the United Kingdom Software Metrics Association (UKSMA) 'The distinction between maintenance activity of minor enhancements and development activity of major enhancement is observed in practice to vary between organizations. The authors are aware that in some organizations activity as large as to require 80 workdays is regarded as maintenance, while in others the limit is five days. Initially it is proposed that the ISBG and UKSMA will adopt the convention that work requiring five days or less will be regarded as maintenance activity'.[*Isb04*]

Bennett [*Ben00*] states that software maintenance requires a number of additional processes and activities not found in software development: a) Modification Requests are usually made to a "help desk" (often part of a larger end-user support unit), which must assess the change; b) Impact analysis and the need for software comprehension; and c) the specialization in regression testing of software so that the new changes do not introduce errors into the parts of the software that were not altered.

In conclusion, software maintenance has unique processes and activities that are not present in the software development domain. Software maintenance also calls on some specific software development processes particularly the implementation processes of ISO/IEC 12207 [*Iso95 s5.5.3*]. Victor Basili states that software maintenance is a specific domain of software engineering, and that it is therefore necessary to look into its processes and methodologies to take into account its specific characteristics [*Bas96*].

2.3 Software maintenance categories

Lientz & Swanson initially identified three categories of maintenance: corrective, adaptive, and perfective. [*Lie78*]. These have been updated, and the International Organization for Standardization (ISO) has defined a new category in the Standard for Software Engineering-Software Maintenance [*Iso98*]. The categories of maintenance defined by ISO/IEC14764 are as follows:

- Corrective maintenance: Reactive modification of a software product performed after delivery to correct discovered problems;
- Adaptive maintenance: Modification of a software product performed after delivery to keep a software product usable in a changed or changing environment;
- Perfective maintenance: Modification of a software product after delivery to improve performance or maintainability;
- Preventive maintenance: Modification of a software product after delivery to detect and correct latent faults in the software product before they become effective faults.

The ISO/IEC14764 international standard of software maintenance classifies Adaptive and Perfective maintenance as enhancements. It also groups together the corrective and preventive maintenance categories into a Correction category, as shown in Figure 1. Preventive

maintenance, the newest category, is most often performed on software products where safety is critical.

Correction	Enhancement
Preventive	Perfective
Corrective	Adaptive

Figure 1: ISO/IEC14764 software maintenance categories

2.4 Software maintenance standards

Software maintenance (refer to figure 2) is one of the five primary processes in the software life cycle as described by the ISO/IEC12207 international standard [*Iso98*, s5.1]. The primary processes of this standard may call on 1) other primary; 2) supporting and; 3) organizational processes when needed. This standard clarifies which of the activities that are also used by the developers, should be used by maintainers (i.e. documentation, configuration management, quality assurance, verification, validation, reviews, audits, problem resolution, process improvement, infrastructure management, and training). [*Moo98*, Fig. 36-37, *Table 94*].

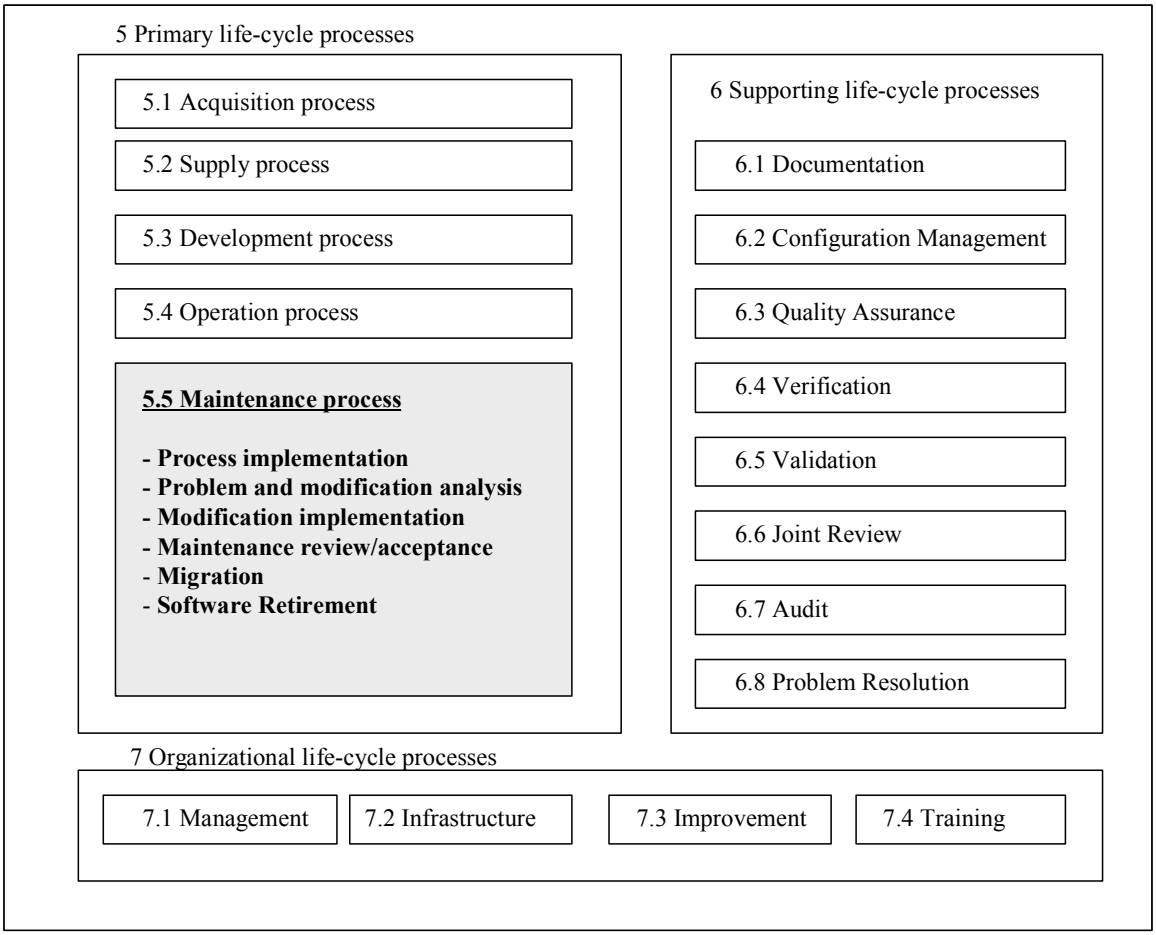

Figure 2: Software maintenance as a primary process of ISO/IEC 12207 [*Iso95*]

An important number of standards apply to software engineering [*Moo98*]. To identify specific standards that apply to software maintenance the key IEEE and ISO/IEC standards and their

relationships have been investigated. A good number of software engineering standards that refer to the software maintenance activities were found [*Mag97 p.325*].

The ISO/IEC12207 international standard describes the overall life cycle of software and is a good starting point before considering more specific standards. It is therefore an excellent document to obtain an overall view of the maintenance process and its relationships with software development. For this research, two standards were found to directly address software maintenance: IEEE1219 and ISO/IEC14764. As introduced in section 2.2 it is stated in the ISO/IEC 14764 and ISO/IEC 12207 that maintainers use some development activities and that they adapt them to meet their specific needs [*Iso98a, s8.3.2.1 and s8.3.2.2*]. More specific references between ISO/IEC 14764 and ISO/IEC 12207 are: a) paragraph 5.5.3.1 of ISO/IEC14764 prescribes to use and adapt software development processes when a need to modify the software arises during its maintenance; b) there is another reference that prescribes that software development processes be used and adapted for documenting testing and review activities (*ISO/IEC 12207 clause 5.5.3.2*).

According to Bennett [*Ben00*, s9.3] software maintenance standards presented by IEEE and ISO/IEC are "*classical or basic*" and do not address newer approaches and processes found on the market today: e.g. Xtreme maintenance [*Jef00*, *Poo01*], '*user computing*' and *Service Level Agreement*s. He states that the current process model used in the software maintenance standards corresponds approximately to level two of the five SEI/CMMi capability maturity levels.

In conclusion, in software engineering there are a large number of standards. Three of them are central to software maintenance: ISO/IEC14767, IEEE1219 and ISO/IEC 12207. These standards indicate that software maintenance refers to software development activities in very specific areas and maintainers must ensure that they adapt them to their specific needs.

2.5 Software Maintenance problems

It is fair to say that software maintenance is not very present in the teaching curriculum of our schools [*Car92*]. The result is a lack of software maintenance culture, knowledge, available techniques and tools for the employees that work in this field.

Problems can either be perceived from an external or internal perspective. It is said that an employee has an internal perspective while the users and customers have an external perspective. [*Dek92*] Dekleva presents a survey report, From the perspective of a software maintenance employee, that lists 19 reported key problems of software maintenance (see table II). The survey participants were attending successive software maintenance conferences over several years.

Table II

Survey on software maintenance problems perceptions [*Dek92*]

Rank	*Maintenance problem*
1	*Follow changing priorities*
2	*Inadequate testing techniques*
3	*Hard to measure performance*
4	*Software documentation incomplete or missing*
5	*Adapt to rapid change of user organizations*

6	*Large backlog or requests*
7	*Hard to measure/demonstrate software maintenance contribution*
8	*Low morale because of lack of respect and understanding*
9	*Few maintenance professionals with experience*
10	*Little methodology; few standards, procedures, or specific tools*
11	*Source code of existing software is complex and not structured*
12	*Integration, overlap, and incompatibility of existing systems*
13	*Low level of training for maintenance staff*
14	*No strategic plans for software maintenance*
15	*Hard to understand and respond to end-user requests*
16	*Little understanding and support from IT management*
17	*Software of systems under maintenance operating on obsolete environments*
18	*Little intention to reengineer existing software*
19	*Loss of expertise when employees leave the team*

It has been widely published that software maintenance is, by itself, of major economic importance. A number of surveys over the last 15 years have shown that for most software, software maintenance occupies anything between 40 and 90 percent of total life cycle costs as published by Foster & Munroe [*Fos87*], 75% according to Rand P. Hall [*Hal87*], 50-80 % according to Tony Scott [*Sco88*] and more than 60% according to Hanna [*Han93*]. This type of survey confirms the users perception that maintenance costs are high. But are the sources of maintenance costs really known by users? And if they are known are they the same as perceived by the maintenance staff?

Jones [*Jon91, Pig97 s2*] describes that this high cost perception originates from a lack of management. He states that software maintenance managers do not communicate adequately all the work that is carried out for their users and customers. Software maintenance managers often regroup enhancement and corrections in the same statistics, budgets and management reports. This perpetuates the notion that most maintenance work is corrective and does not clearly represent the importance and added value of the other maintenance categories.

'*The more substantial portion of maintenance cost is devoted to accommodating functional changes to the software necessary to keep pace with changing user needs. Based on data reviewed, it was also noted that systems with well-structured software were much better able to accommodate such changes.* [*For92*]'.

Lientz and Swanson [*Lie80*] presents, on the basis of a survey of 487 software maintenance organizations, that 55% of requests are, in fact, new requirements as opposed to corrections. In questioning software maintenance personnel, Pressman [*Pre97, s27.2.1*] finds similar results with 50% to 80% of maintenance effort dedicated to adding new functionality required by the users and to answer all kinds of support enquiries concerning the business rules of operational software. Since this fact is not well communicated to users and customers, the perception is still that maintenance is mainly corrections of failures.

Another misconception that users have is that hardware and software should have comparative maintenance costs and efforts. In fact this is not the case. 'Hardware degrades without maintenance while software degrades because of maintenance activities [*Gla92*].'

Pigoski [*Pig97 s2*] states that software maintenance is labor intensive and that the majority of costs are associated with the human resource component of the cost. Because of economies of scale and new production processes, the hardware no longer accounts, for the majority of the costs of modern software systems. The manpower costs (*e.g. at Cable & Wireless during 2003 a cost of $1,500 USD per day for a SAP/r3 Abap programmer*) are now at the top of finance and management preoccupations. It then becomes very important to explain clearly and provide details of where time is spent during software maintenance. This is crucial to enhance the perception of the value of the many services.

Banker [*Ban93*] states that the size and complexity of software greatly influences its maintenance costs and modification efforts. To such extent that Boehm [*Boe87*] publishes that for each dollar invested in developing software there will be a spending of 2 dollars in maintenance. He continues by explaining that maintenance costs can be stated as a function of the number of instructions in the source code of the software.

Lehman [*Leh85, Leh97*] also indicates that the structure of the software code, which undergoes successive maintenance activities, becomes progressively more complex because of the many changes. As a result a growing number of human and other resources need to be assigned to maintain software that becomes more and more complex over time. This argument is not fully supported by the software maintenance literature. Pigoski [*Pig97*] observed larger number of changes during the first three years of maintenance of new software and recommends that management should allocate more maintenance resources during the first years in service and after that time, progressively fewer resources will be required as the software stabilizes and gradually becomes obsolete.

Osborne and Chikosky [*Osb90*] blame the age of operational systems for the complexity. They argue that the average age of operational software is from 10 to 15 years. They present the point of view that in the past the software community did not have access to modern architectural techniques. These old software, also called legacy software, demonstrate a more complex internal structure, bad coding practices and weak documentation which all contribute to higher maintenance costs and efforts.

Finally, other research work at Hewlett Packard [*Hp90*] identifies that the main factors, in decreasing importance, that contribute to high costs of software maintenance in their company are: the number and the experience of the programmers, the quality of the technical documentation and user documentation, the tools used by the maintenance employees, the structure and maintainability of the software and last, the contractual obligations that constrains the maintenance activities.

2.6 Software Maintenance Context

It is important to further explain and describe the scope of software maintenance activities and the context in which maintainers work daily (see Figure 3). There are indeed multiple interfaces in a typical software maintenance organizational context where the maintenance manager must keep his applications running smoothly. He must react quickly to restore order when there are production problems. He must provide the agreed-upon level of service. He must keep the user-community confident that they have a dedicated and competent support team at their disposal, which is acting within the agreed-upon budget" [*Abr93*].

The interface with the user is a key function and relates to the daily communications which require: a) rapid operational responses to problem reports; b) responsiveness to inquiries about a specific business rule, screen or report; and c) progress reports on a large number of modification requests.

Such user interfaces are either direct, or accessible via a Help Desk, and, in best practices, are supported by a ticket-handling system, which documents, controls and expedites the workload. Other user interface activities, less intense and less frequent, consist of negotiations and discussions about individual request priorities, service level agreements (SLAs), planning, budgeting/pricing and user satisfaction-related activities.

A second maintenance interface deals with a) the Help Desk; and b) the infrastructure and operations organization [*Iti01a, Iti01b*]. The user is rarely aware of, or involved in, the details of software engineering processes. He is unaware of the many daily interactions between these two organizations. Internally software engineers must have an effective problem resolution process and efficient communications to ensure quick and effective resolution of failures.

A specific request, sometimes called a "ticket" when this process is automated, will typically circulate among software engineer support groups in order to isolate a problem [*Apr01*]. The user interface also includes less frequent activities such as coordination of service recovery after failures or disasters in order to help restore access to services, within agreed-upon SLA terms and conditions.

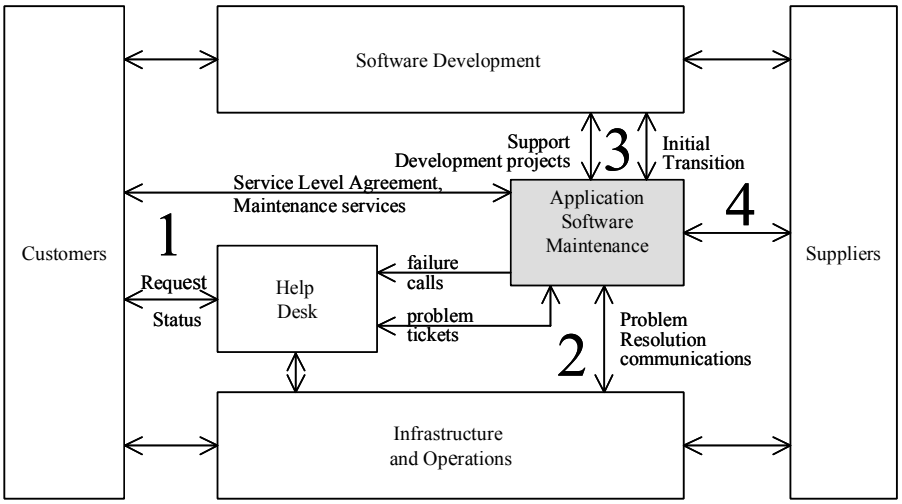

Figure 3: Software Maintainers Context Diagram

The third key interface exists between the software developers and the software maintainers, and is initiated during the development of new software. The root cause of several maintenance problems can be traced to development, and it is recognized that the maintainers need to be involved and exercise some form of control during this transition [*Dek92, Wal94, Pig97, Ben00*]. This development-maintenance interface also illustrates the contributions made by maintainers to help in and support, and sometimes be involved in, a number of large development projects concurrently. The maintainer's knowledge of the software and data portfolios is of great value to the developers, who need to replace or interface with legacy software. Some of the key

activities would be, for example: a) development of transition strategies to replace existing software; b) help in the design of temporary or new interfaces; c) verification of business rules or help in understand the data of existing software; and d) help in data migration and cutover of new software or interface.

The fourth interface (in figure 3) addresses relationships with a growing number of suppliers, outsourcers, and ERP vendors [*Car94, Apr01, McC02*]. The maintainers interface with them in all kinds of relationships, for example: a) with suppliers that develop new software or configuring ERP software; b) with sub-contractors who are part of the maintenance team, to help with specific expertise and additional manpower during peak periods; c) with suppliers of maintenance contracts providing specific support services for their already licensed software; and d) with outsourcers who might partially or completely replace a function of the software engineering organization (*development, maintenance or operations & infrastructure*). To ensure good service to its user, software maintainers must develop some understanding of the many contract types, and manage them efficiently, to ensure supplier performance, which often impact the SLAs results.

2.7 Software Maintenance Processes and activities

Authors report that many software organizations do not have any defined processes for their software maintenance activities [*Pia98*]. Van Bon [*Van00*] confirms the lack of process management in software maintenance and that it is mainly a neglected area. What is the source of this lack of interest in process and procedures? Schneidewind [*Sch87*] tells us that, traditionally, maintenance has been depicted as the last activity of the software development process. This can still be seen today in the IEEE1074-1997 standard, which represents software maintenance as the seventh step of eight software development steps. Even today, many industrial software engineering methodologies do not even represent the software maintenance processes or activities [*Sch00*]. As an example the British Telecommunications software development methodology presents maintenance as a unique activity at the end of the software development [*Btu90*]. Bennet [*Ben00*] has an historical view of this problem and traces it back to the beginning of the software industry, where there was no difference between development and maintenance of software. Differences only started to appear during the 1970's when software maintenance life cycles started to appear. He describes that the first software maintenance life cycles had three simple activities: 1) comprehension; 2) modification, and 3) validation of the software change.

The 1980's brought more extensive software maintenance process models [*Ben00, Iti01a, Iti01b, Fug96*]. In these life-cycle models, software maintenance is not represented as the last stage of software development. These models present specific software maintenance activities and introduce a sequence for each activity. Some consulting firms define their own maintenance life-cycle models offering specialized activities for their markets and customers. The many proposals culminate in the development of national and international standards in software maintenance during 1998 with the publication of IEEE 1219 [*Iee98*] and ISO/IEC14764 [*Iso98*] that are currently in use today.

As a first step to identifying all the key software maintenance processes and activities, these two standards have been used to develop a detailed list of software maintenance processes and activities (see Annex A).

This first inventory is useful in structuring maintenance processes and activities and presenting the current scope and structure of key software engineering publications.

2.7.1 Unique software maintenance processes and activities

Depending on the source of the maintenance requests, maintenance activities are handled through distinct processes; this is illustrated in Table III with a few examples. For each request source, a key maintenance service/process, together with due registration of the related maintenance categories of work, is initiated. For example, if users are the source of the requests, then a change request related to operational use of the software and the work to be carried out can be classified within one of three maintenance services: correction, evolution or operational support. In some instances, a supporting process will be needed. A typical one is the need for service level agreement information as part of the operational support activities.

Table III

Activities and Categories of maintenance work

Source of Requests	Example of a Key Maintenance Service/Process	Assignment to a Maintenance Category for maintenance effort collection
Project Managers	Management of transition from development to maintenance	Operational Support for project
Project Managers	Provide knowledge of existing legacy systems	Operational Support to project
Users	Ask for a new report or complex query	Operational Support to users
Users	Ask for new functionality	Adaptive
Users	Report an operational problem	Corrective
Users	Quarterly account management meeting with the users	Operational Support to users and SLA
Software Operations	Change to a systems utility	Perfective
Rejuvenating Studies	Software impact analysis	If large enough, it can be assigned to preventive maintenance, and often leads to a project or to redevelopment, both of which are outside the scope of small maintenance activities.

A list of distinct software maintenance processes can be found in the recent version of the Software Engineering Body of Knowledge (SWEBOK) which identifies a number of processes and activities that are unique to maintainers, for example [*Abr01*]:

- Transition: a controlled and coordinated sequence of activities during which a system is transferred progressively from the developer to the maintainer [*Dek92, Pig97*];
- Service Level Agreements (SLAs) and specialized (domain-specific) maintenance contracts [*Apr01*] negotiated by maintainers;
- Modification Request and Problem Report Help Desk: a problem-handling process used by maintainers to prioritize, document and route the requests they receive [*Ben00*];

- Modification Request acceptance/rejection: modification request work over a certain size/effort/complexity may be rejected by maintainers and rerouted to a developer. [*Dor02, Apr01*].

While doing the inventory of maintenance activities and literature review, it was confirmed that some maintenance processes and activities are unique to maintainers and are not present in the software development function (see Table IV).

Table IV

Software maintenance key process areas (P = present, A = absent)

Some Maintenance Key Processes	Software management (maintenance)	Software development (creation)
Management of problems (Problem resolution interfacing with a help desk)	P	A
Acceptance of the software	P	A
Managing transition from development to maintenance	P	A
SLAs	P	A
Maintenance planning activities (versions, SLAs, and impact analysis)	P	A
Event and service request management	P	A
Software management (operational support)	P	A
Software rejuvenation	P	A

A number of software engineering topics and sub-topics have also been found to be adapted for the specific nature of software maintenance, including:

Process Simulation: Process simulation techniques are used in the maintenance area. These techniques are used for improvement activities to optimize the maintenance processes and case studies are described in [*Bar95*].

Software Maintenance Measurement: Maintainers extensively use satisfaction surveys to understand how their customers are doing [*But95*, *Cfi02*[2]]. Maintainers use internal benchmarking techniques to compare different maintenance organizations and products to improve their processes internally [*Abr93a*, *Bou96a*]. External benchmarking of software maintenance organizations is now becoming more popular [*Abr93a, Ifp94*, *Mai02*, *Isb04*]. Measurement initiatives specific to maintainers are also described in publications [*Gra87*, *Abr91*, *Abr93*, *Stp93*, *Sta94*, *Mcg95*]. Software estimation models specific to maintenance have also been published [***Bas79***, ***Nie88***, *Abr95*, ***Gef96***, ***Hen96***, ***Nie97***]. Pressman [*Pre97 paragraph 4.5.2*] also indicates that no one measure can be found to reflect the maintainability of software, and that a number of indicators are required! This leads to external and internal measurement of the maintainability of software done by some organizations using commercial tools [*Boo94*, *Lag96*, *Apr00*].

Maintenance Request Repository: An adequate information system (often shared with the operations help desk area) must be used by the maintainer to manage the workload and to track a large number of users' requests. It can become the basis for the effort collection and an

[2] Provided as an example of a satisfaction survey consulting firm

important component of the measurement infrastructure [*Gla81, Art88, Iti01a paragraph 4.4.7, Kaj01d, Nie02 activity 3*].

Software Maintainer specific training and education: the following references on maintainers training and education address the specifics for software maintainers [*Kaj01c*, *Kaj01e*, *Kaj01f*, *Hum00*, *Pfl01 paragraph 10.1 and chapter 11*]

Billing of the maintainers' services: More and more often, the maintainer must accurately track his work and issue a maintenance billing to the customer organization. This must, of course, be supported by the development of a billing policy [*Iti01a, 5.4.2*]. Maintenance service items and prices must be clarified and supported by a software maintenance billing process and supporting systems.

Production systems surveillance: A maintenance organization must also put in place a production system surveillance set-up to probe, every day, the operational environment for signs of degradation or failures. Such surveillance systems ensure that problems are identified as early as possible (hopefully before the user is aware of it) [*Iti01a paragraph 4.4.8*].

2.8 A classification of the software maintainer's key processes

Taking into consideration the list of processes and activities a high-level process model of software maintenance activities was developed. It presents key software maintenance processes grouped into three classes (Figure 4). The main idea is to provide a similar representation used by ISO/IEC 12207 standard but focused on software maintenance-specific processes and activities:

a) Primary processes (software maintenance operational processes);
b) Support processes (supporting the primary processes); and
c) Organizational processes offered by the software engineering organization or by other organizations (for example: finance, human resources, purchasing, etc.).

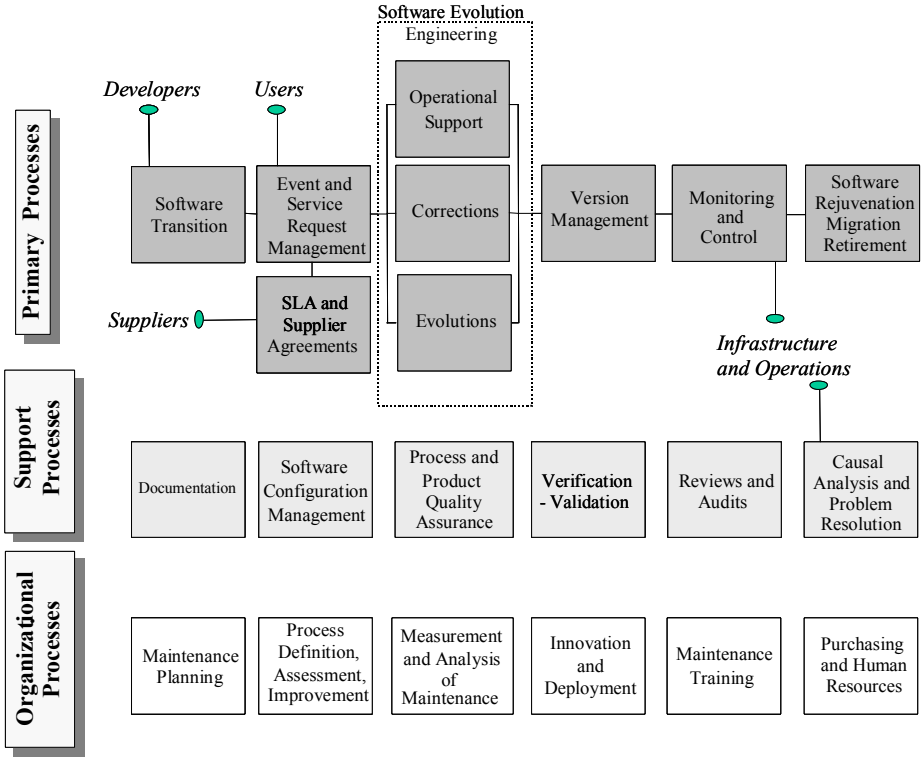

Figure 4: A classification of the Software Maintainer's Key Processes

The key operational processes (also called primary processes) that a software maintenance organization uses are initiated at the start of software project development beginning with the transition process. The transition process is not limited, as some standards present, to the moment that developers hand over the system to maintainers. 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. It means that a process is implemented to follow the developer during the system development life cycle. 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, rerouted, or rejected (on the basis of the service-level agreement and the nature of the request and its size)[*Apr01*]. 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*. Note that certain service requests do not lead to any modification to the software. In the model, they are referred to as 'operational support' activities, and these consist of: a) replies to functionality questions; b) provision of information and counselling; and c) helping customers to better understand the software, a specific transaction, the internal validations, its data or its documentation.

The last two main operational processes concern the *Version Management process*, moving items to production, and the *Production Surveillance process*, ensuring that the operational environment has not been degraded. Maintainers always 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 that needs to be investigated.

A process which is used, when required, by an operational process is said to be an operational support process. This classification includes: a) the many maintenance planning perspectives (i.e. year plan, system plan, impact analysis of a specific request); b) the software configuration management function and tools which is often shared with developers; c) the maintenance and testing environments; d) management of the contractual aspects (i.e. escrow, licenses, third-party) and service level agreements; e) rejuvenation or retirement of software; and, finally, f) the problem resolution process often shared with infrastructure and operations. These are all key processes required to support software maintenance operational process activities.

Organizational processes are typically offered by the IS organization and by other departments in the organization (for example: human resources, finance, and quality assurance). While it is important to measure and assess these processes, it is more important for the maintainer to start defining and optimising the operational processes first. The operational support processes and the organizational processes follow these.

This generic software maintenance process model helps to understand and represent the various key software maintenance processes but it lacks conformance to ISO12207. Another version of the maintenance process model that could be readily accepted by the standards community is presented in figure 5. In this figure the updated process view of software maintenance is highlighted and shows its integration in the existing process model.

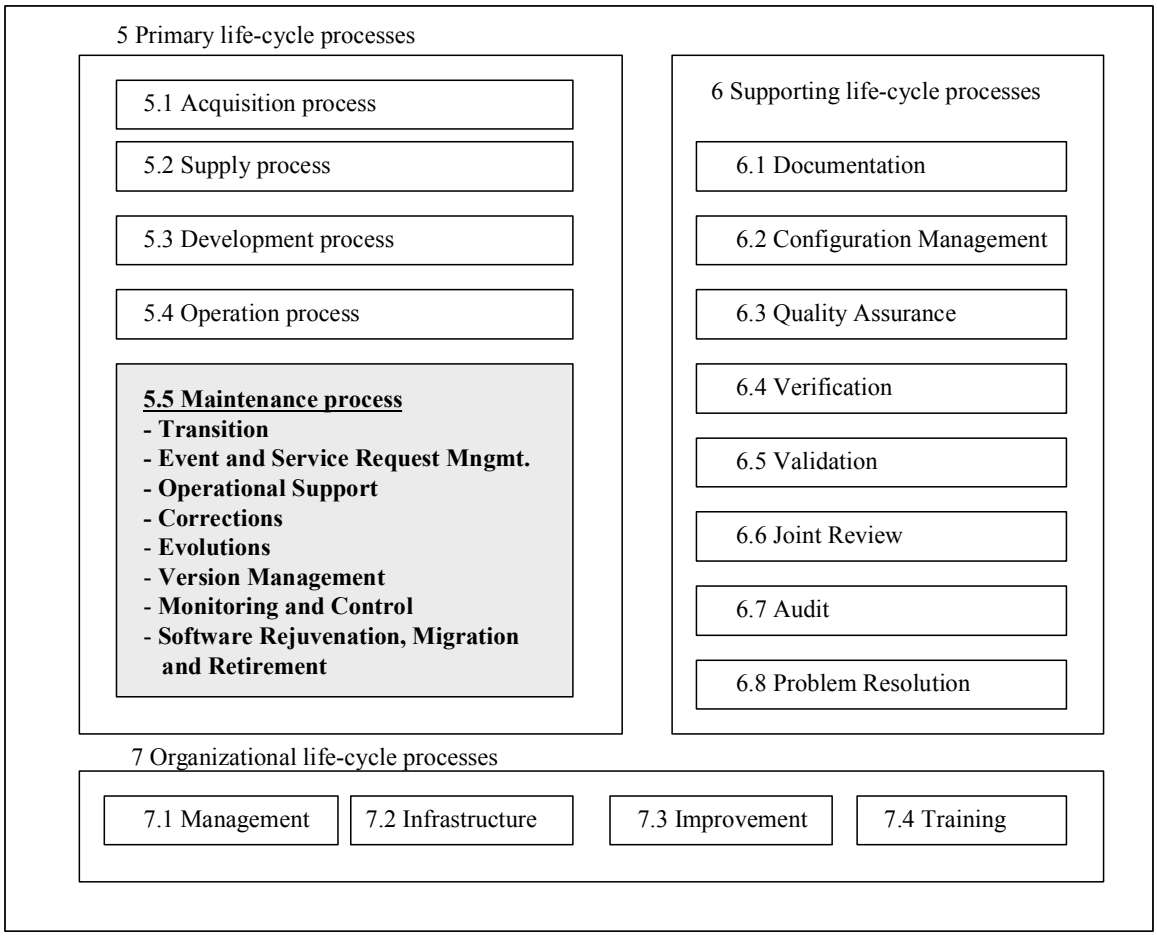

Figure 5: Proposed update to ISO12207 software maintenance processes

3 SOFTWARE ENGINEERING MATURITY MODEL LITERATURE REVIEW

Since the middle of the 1980's, many capability maturity models have emerged. The concept has become so popular that many other industries are developing capability maturity models, for example human resources management [*Aps01*], financial management [*Des99*] and primary care [*Gil99*].

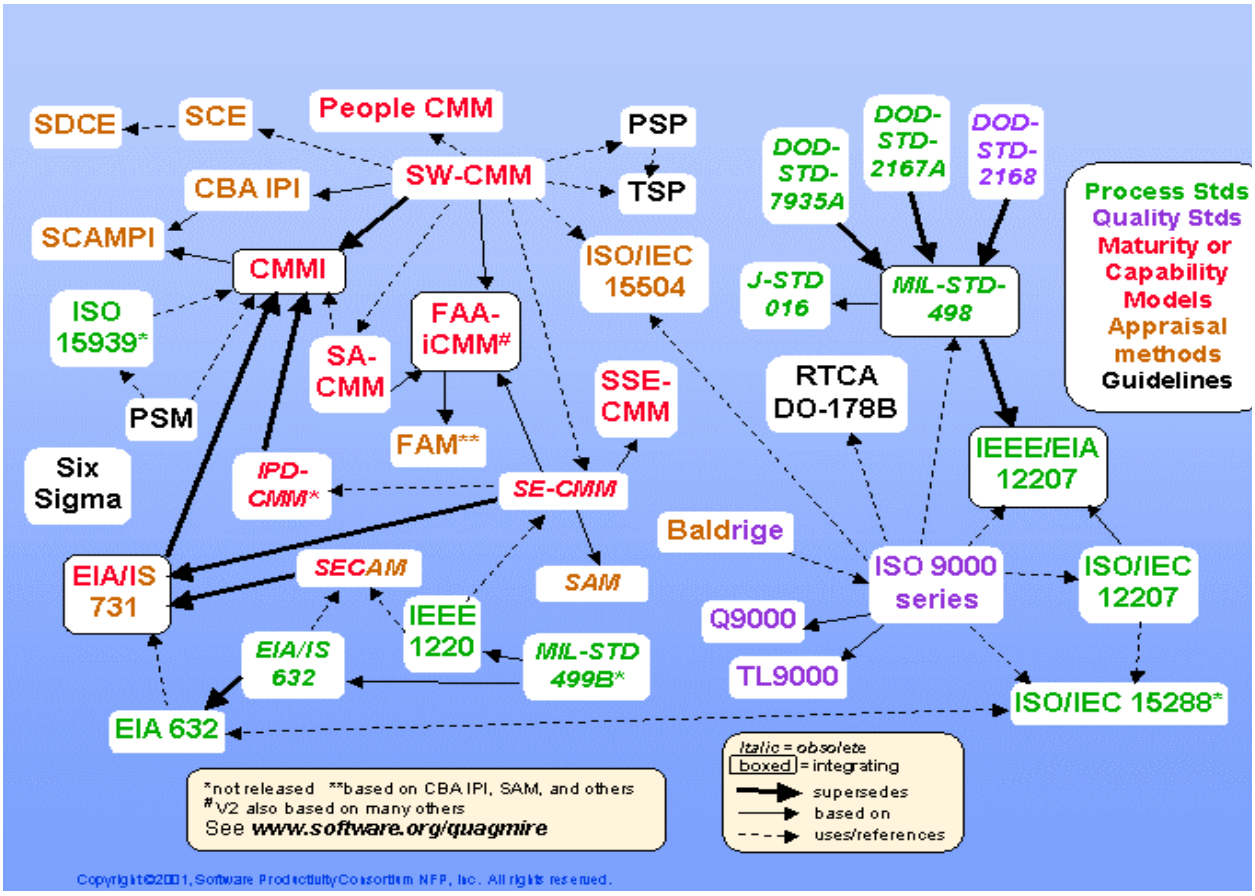

Figure 6: 'The Frameworks Quagmire' [*She01*][3]

Sheard [*She01*] presents (see figure 6) the most popular capability maturity model issue from the USA. Moore [*Moo98*] complements this list with European and Canadian models while El-Eman and Brito and Abreu [***Ema98, Bri99***] have published an historical view of this domain. A short description of the models is presented in table V. Each model identifies the date of publication, the origin and an indication of whether the details are public domain or not.

These models are the most published. There is not, in this list, a capability maturity model specific to software maintenance. Maybe a proposal for a comprehensive maintenance-specific capability maturity model exists somewhere else? A literature search has not resulted in any comprehensive diagnostic techniques to evaluate the quality of the maintenance process, as described by the high level process model of figure 4.

[3] Used with the permission of Sarah Sheard (sheard@software.org)

Table V

History of the most popular software related maturity models

Model name	Origin	Public domain	Description
Maturity Framework		Yes	Watts Humprey publishes in 1987 a first paper of concepts of process maturity developed by the SEI. His book 'Managing the Software Process' is published in 1989.
SW-CMM		Yes	Maturity Model (Published in 1991) for software improvement and assessment.
Trillium et Camélia	Bell Canada	Yes	The model (Published in 1991) by Bell Canada and Nortel integrates the SW-CMM, ISO9001 and other standards to assess the development and maintenance of telecommunications products.The last version of Trillium was published in 1996 (v3.0). Camélia, is a french version of Trillium and improves the model that now integrates MIS (Management Information Systems) concepts. It is published in 1994.
Process Advisor	Dr. Pressman	Yes	Pressman, during 1992, publishes a list of questions questions that cover eight capability domains. It can be considered a simple proposal for a maturity model that benchmarks industry practices with best practice. [Pre92]
Bootstrap	ESPRIT R&D projects	No	'The European software capability and process improvement model (Published in 1993) has been developed in a research initiative named ESPRIT #n5441. Bootstrap, SPICE, ISO9001 and the CMM ® are used in the mapping of this maturity model. The model is developed and maintained by the Bootstrap Institute of Bruxel. Last version is 3.0 in year 2000'
SQPA	HP	No	Model (Published in 1993) by the company Hewlett Packard that uses the research work of Capers Jones. [Ibr03]
FAA-ICMM	FAA	Yes	This model (Published in 1993) integrates the SW-CMM,the System Engineering CMMand the Software Acquisition CMM. It is intended for use in the Federal Aviation Administration of the USA.
STD	Compita	No	Model (Published in 1994) by the Scottish Enterprise is only accessible through a licence agreement with the Compita society. The model claims conformance to SPICE ® and uses a proprietary evaluation method named PPA.
SAM	BT	No	Model (Published in 1994) by British Telecommunications for internal assessments of its software processes.
SE-CMM	EPIC	Yes	Model (published in 1995) describes key practices of a system engineering organization.
People CMM	SEI	Yes	Model (Published in 1995) describes key elements of HR development and management in software development and software engineering driven organizations.
SSE-CMM	ISSEA	Yes	Model (Published in 1996) describes the caracteristics of the processes of software engineering in a security perspective.
IPD-CMM	EPIC	Yes	Model (Initiated during 1996) to improve the processes of the whole life-cycle of all the processes involved in the development of a product (including software). The project did not finish and was stopped in 1997.
SA-CMM	SEI	Yes	Model (Published in 1996) with an architecture similar to the CMM model, was designed for purchasing software.
SECAM	INCOSE	Yes	Model (Published in 1996) based on a questionnaire with the objective of assessing the evolution of existing system engineering practices.
(SPICE) 15504	ISO/IEC	Yes	Technical reports from ISO/IEC (published in 1998) with the number 15504 contains models and assessment methods for improvement and

			capability assessment of software processes. A reference model is included. Ten reports are published.
EIA/IS 731	EPIC-INCOSE	Yes	Model (Published in 1999) integrates SE-CMM and SECAM giving SECM. This model is based on system engineering principles of (EIA/IS 731).
CMMi	SEI	Yes	Version 1.1 (Published in 2002) integrates many of the SEI software process models. This new version renders obsolete many previous models from the SEI.

Table VI presents an inventory of recent proposals of software engineering process evaluation and assessment models. Each of these models was analyzed to identify contributions that could help maintainers. Of the thirty-four proposed models in this review, only a handful (shown in **bold** in table VI) includes documented maintenance practices, sometimes accompanied by a rationale and references. However, none of these models covers the entire set of topics and concepts of the process model presented earlier (see figure 4).

Table VI

Software Engineering Maturity Models proposals, (*sorted by year of publication*)

Year	Software Engineering Maturity Model proposals
1991	Sei91, Tri91, Boo91
1993	Sei93
1994	**Cam94**[4], Kra94
1995	Cur95, **Zit95**
1996	Bur96 & Bur96a, Dov96, Hop96, Men96
1997	Som97
1998	Top98, Baj98, Ear98
1999	Wit99, Vet99, Sch99, Faa99, Gar99
2000	Str00, Bev00, Lud00, Luf00, **Cob00**
2001	**Kaj01d**, **Kaj01c**, Ray01, Sri01
2002	**Sei02**, **Nie02**, Mul02, Vee02, Pom02, Raf02, Sch02, Ker02, Cra02, Win02
2003	Nas03, Doc03, Sch03a, Wid03, Rea03

Using these proposals the first inventory of software maintenance processes and activities of annex A is enhanced. The result of this activity is a much more comprehensive list of software maintenance processes and activities (see Annex B) that covers: a) national and international standards; b) relevant software maintenance-specific capability maturity model proposals; and c) recognized key software maintenance references.

From these two successive mappings, a large number of software maintenance best practices have been identified and listed. To summarize, the key software maintenance references that should be used to develop a comprehensive maintenance-specific capability maturity model (***SM***^{mm}) are:

- ISO/IEC14764 [*Iso98*];

[4] Cam94 includes and expands on Tri91 detailed practices

- IEEE1219 [*Iee98*];
- ISO/IEC12207 [*Iso95*]
- The CMMi© [*Sei02*];
- SWEBOK [*Abr01*]

The revised ***SM^{mm}*** model has also taken inputs from, and makes references to, other capability maturity models and best practices publications that consider a variety of software maintenance-related topics:

- *Camélia* Capability Maturity Model [*Cam94*];
- Model to improve maintenance of software [*Zit95*].
- CobIT [*Cob00*];
- Cm3-Corrective Maintenance Model [*Kaj01a*];
- Cm3-Maintainer's Education Model [*Kaj01b*];
- IT Service CMM [*Nie02*];

This list was used to survey 35 software maintenance specialists and ask them if this set of document is complete and well suited to represent the software maintenance knowledge area. Many of the respondents supplied us with the following additional requirements:

- ITIL Service Support [*Iti01a*];
- ITIL Service Delivery [*Iti01b*];
- Malcolm-Baldrige [*Mal04*];
- Iso90003:2004 [*Iso04*]
- Process evaluation model standard ISO/IEC TR 15504 (SPICE) [*Iso98a*];

3.1 CMM© and CMMi© models limitations

There is still a strong view that there is no need for a software maintenance-specific capability maturity model because the CMMi© claims to be addressing this topic in enough detail. This article confirms that some maintenance processes are unique to maintainers and are not part of the software development function (see table III). This means that the SEI currently views software maintenance as a project which it is not the case for small enhancements. When these unique maintenance processes are compared to the CMMi© model content, it can be observed that the CMMi© model, being highly centered on the software development, does not explicitly address these topics, nor, with its primary focus on project management, does it explicitly address the issues specific to the software maintenance function [*Zit95, Apr03*]. For example, in the CMMi©:

- The concept of maintenance maturity is not recognized or addressed;
- There is no sufficient inclusion of maintenance specific practices as process improvement mechanisms;
- Maintenance-specific issues are not adequately addressed;
- Rejuvenating-related plans such as need for re-documentation, re-engineering, reverse engineering, software migration or retirement are not satisfactorily addressed.

This was also observed in the previous version of the model, the CMM©, in 1995 [*Zit95*] and still absent from the new CMMi© version, since it maintains a developer's view of the software production process.

4 RESEARCH QUESTIONS

This article investigates the following research questions:

a. What are the processes and activities as well as the unique activities of software maintenance?
b. Software maintenance refers to the software development standards. Can the standards and activities that are shared be identified clearly?
c. Are the unique processes of software maintenance well reflected in the current international standards?
d. Is there a maintenance-specific capability maturity model that covers the entire set of software maintenance specific processes?
e. What would be the proposed architecture of a maintenance-specific capability maturity model that could address the entire set of software maintenance unique activities?
f. Can we show examples of detailed practices of such a model?

5 RESEARCH METHODOLOGY

This section presents an overview of the research methodology selected to address the research questions described. The research methodology is composed of four distinct stages (see table VII): Definition, Planning, Development and Validation.

1- The definition stage was composed of the following activities:
- The selection of a dissertation topic which has the potential for the award of a PhD;
- The successful admission in the PhD program;
- The attendance and participation in the program courses;
- The successful completion of the PhD examination on the topic of software maintenance and software engineering capability maturity models.

2- The planning stage of this dissertation project outlines the methodological approach of the research activities as well as the timeline for completion. Initial proposal of the subject as well as two literature review reports [*Apr02a*, *Apr02b*] were presented to the ÉTS during on 10-11-2002 and 30-12-2002. A first plan was presented to Dr. Abran of ÉTS on January 19th 2004 [ ].

3- The development and operation stage of this project consisted in an extensive inventory of software maintenance activities (see appendix 2) followed by successive mappings of the many source documents. Mappings consist of taking detailed source documents one by one and allocate each practice into the key process areas and roadmaps of each maturity levels of the new model. This activity led to the definition of the architecture of the SMmm model which is a software maintenance-specific capability maturity model. This was followed by a number of validation activities that have been carried out to conduct an initial validation of the model concepts including a number of publications made to software engineering conferences and journals. In such a concept research activity it is desirable to illustrate how the new maintenance-specific capability maturity model can be used to explain and understand the current maturity level and potential improvement paths. To be exhaustive this would require numerous field trials of the proposed model. For this research to be achievable in a reasonable time only initial validation of the model has been planned.

   Initial validation was performed in an industrial trial under the sponsorship of the Bahrain Telecommunications IS Planning Director. During the years 2001-02 the model was used in four process appraisals of small maintenance units (6 to 8 individuals) of this organization. Results rated three of the maintenance units as Level 1 and one as Level 2. Following this trial, a number of maintenance specialists and managers were asked how they perceived the assessment process, the reference model, and the assessment results. The results identified the need to include software maintenance practices from the quality perspective, such as Malcolm Baldrige, ISO9000 and the Information Technology Infrastructure Library best practices guidelines [*Iti01*, *Iti01b*]. The SMmm assessments proved useful for the Bahrain Telecommunications Company where improvement activities were immediately initiated in the areas of product measurement [*Apr00*] and SLAs [*Apr01*]. Other validation activities comes from the reviews and acceptance by peers of ISO/JTC1, SWEBOK and many other conferences and journals of the concepts presented in the new model.

4- Refer to section 8 of this article for the interpretation of the contribution of this research.

Table VII
Methodological approach of the research project

Stage 1: Definition

Motivation	Objective	Proposal	Research Users
Improve software maintenance activities in the industry.	Support software maintenance management and personnel improvement efforts.	Develop a capability maturity model specific to software maintenance.	• Software maintenance managers; • Software engineers specialized in software improvement; • Researchers in software engineering.

Stage 2: Planning

Planning Activities of the project	Inputs	Outputs
Initial investigation: 1- Software maintenance litterature review and inventory of the detailed practices of software maintenance. 2- Capability Models literature review and inventory of current CMM proposals. 3- High level schedule and course list	• Software maintenance bibliographic chain • Software engineering capability maturity models bibliographic chain (books, monographs, reports, publications, periodicals, work-in progress, internet, ect)	• One technical report covering the state of the art in those two areas of software engineering [*Apr03a*]; • First inventory of software maintenance processes, activities and practices (see annex A); • Initial mapping of software maintenance practices to CMMi domains architecture.

Stage 3: Development and Operation

Preparation	Execution	Analysis
1- Inventory completion 2- Sucessive mappings of the source documents 3- Design of the Model Architecture and content	• Study of the numerous sources to be used to develop a software maintenance-specific capability maturity model architecture • Develop a mapping process to build the content of the maintenance-specific capability maturity model • Establish the model document table of contents • Map the detailed practices of the source documents to develop a first verion of the maintenance-specific capability maturity model	• High-level architecture of the software maintenance-specific capability maturity model composed of domains, key process areas and roadmaps. • Software maintenance-specific capability maturity model table of contents Fisrt two chapters of the model document: 1) Part 1 of the model presents an overview of the maturity model including its objectives and advantages; 2) Part II of the model describes the model architecture, its scope and maturity levels.

4-Model validation 5- Contribution to the international infrastructure on Software Engineering	• Use with experts and software maintenance organisations • Journals and conferences	• Use of the model in 3 process assessments • Ask opinion of 35 software maintenance professsionnals concerning the model • Publications in refereed journals and conferences • Use of the model to submit enhancements to the SWEBOK project • Use of the model to submit enhancements to the ISO/JTC1 SC7 committe which is reviewing the ISO14764 Software Maintenance International Standard.

Stage 4: Interpretation

Interpretation Context	Extrapolation of results	Futur Work
• This model is an important contribution to the concepts, processes and practices of software maintenance in the context of software process improvement.	• A number of companies and University research laboratories have made requests to participate in this ongoing research project.	• Publication of the model content (french edition) • Development of evaluation tools (ex : questionnaire, assessment tool); • Traceability tables to the source document practices; • Training material for industry assessment team training; • Publication of the model content (english edition); • Proposal to use the results of this research in ISO JTC1/SC7 standards committees.

6 *SM*^{mm} INITIAL MODEL AND ARCHITECTURE

6.1 Identification of process domains and key process areas for software maintenance

Considering the findings of the previous section of this article, the need to develop a software maintenance-specific capability maturity model was now clearer. Then, how to come up with the ***SM***^{mm} model high-level architecture?

After doing the successive inventory of detailed practices (see annex A & B) four process domains were identify to closely match the CMMi architecture (see figure 7).

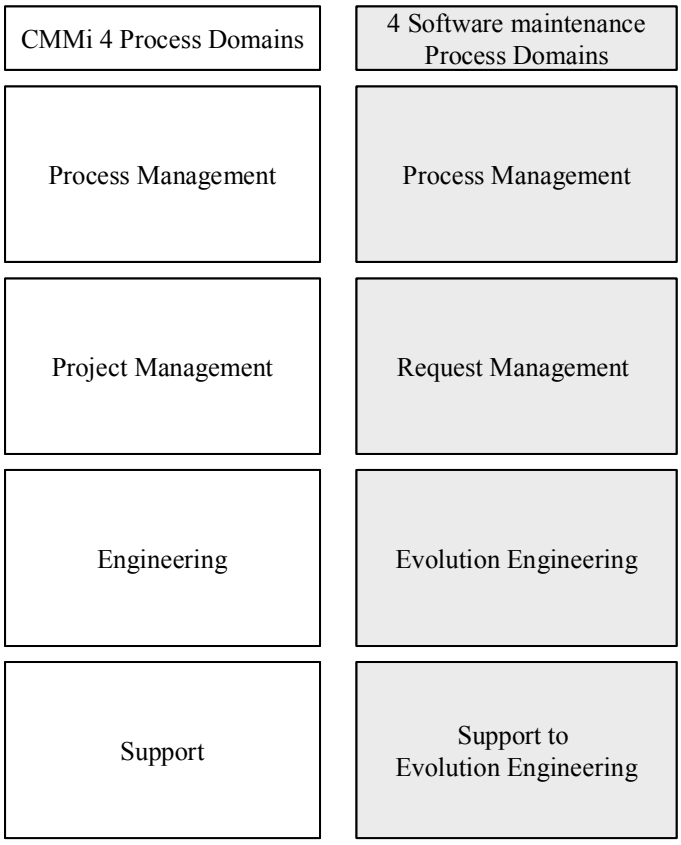

Figure 7: Process domains of software maintenance

A few changes are made when compared to the CMMi. This is intentional. The new model should complement and respect the overall architecture of the CMMi. This is required because most process improvement managers are already familiar with the CMMi and will want consistency in terms of terminology across the models. The most important difference is located in the request management that replaces project management. The reasoning behind this change is that software projects structure and management techniques are not used in maintenance (effort of 5 days or less) but mostly used by development. Project management is necessary for larger initiatives like large maintenance projects. The maintenance projects (which have a project structure and large effort) use the CMMi as it is geared towards project management while ***SM***^{mm} is not.

The next step was to clarify that the engineering processes are adapted for software maintenance work. The term evolution engineering is used for this. The last modification concerns the clarification that support processes will be designed to support the evolution

engineering processes. As a design step the coverage with the processes identified by the ISO/IEC 12207 [*Iso95*] and ISO 14764 [*Iso98a*] international standards (see figure 8) is assessed.

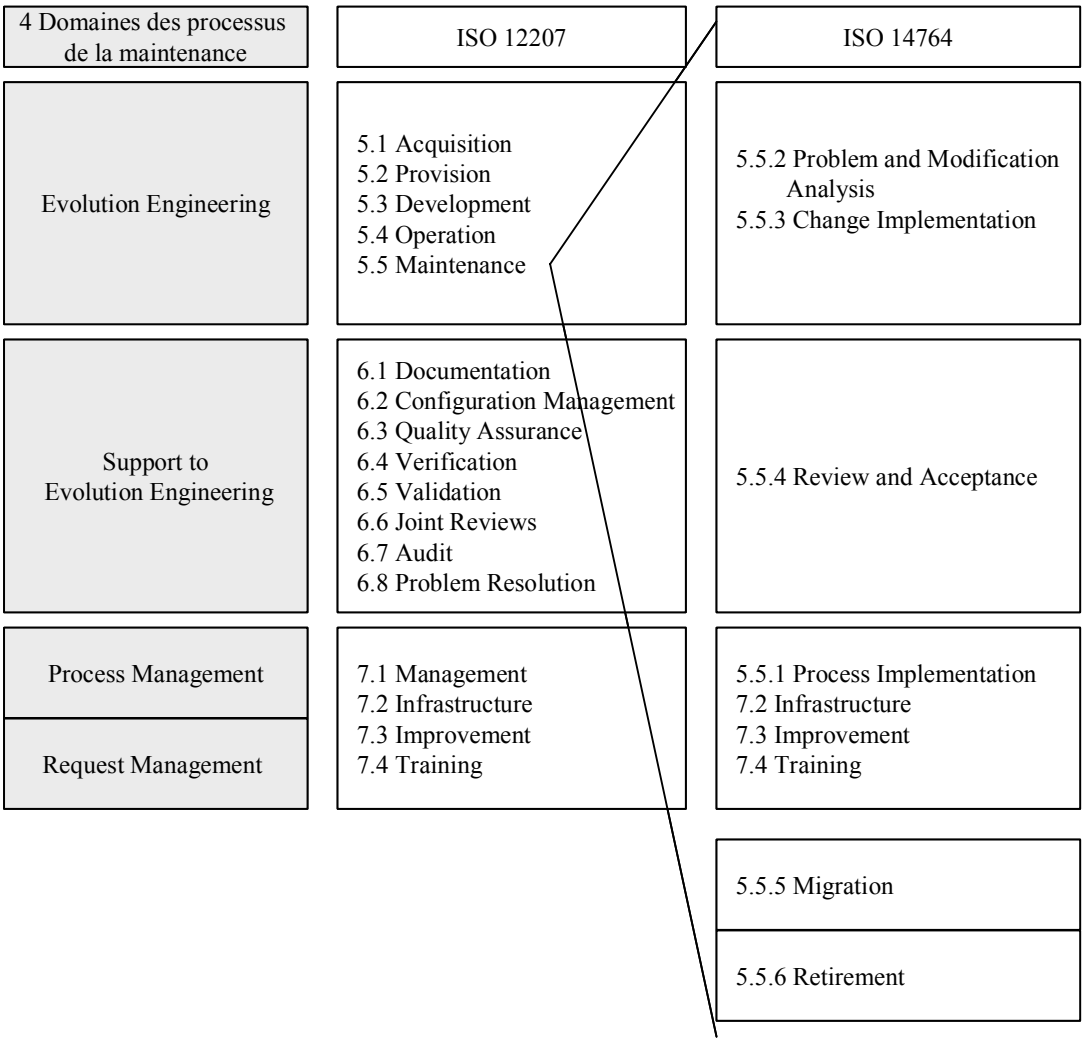

Figure 8: Mapping of the topics of key maintenance standards

This figure shows the mapping of the standards software maintenance processes and what is likely to be included in each one of the domains. The model architecture will need to identify where other primary processes will be referenced, how the support processes of 12207 are likely to be used by the maintainer and what should be contained in process and request management to fully cover the standards.

This last mapping as well as the detailed one presented in annexes A and B has led to the definition of software maintenance key process areas of the proposed capability maturity model.

The first step taken is to re-order the processes to align to the CMMi model: a) process; b) request management; c) evolution engineering and d) support to evolution engineering. The next step was to make an inventory of activities of the CMMi and use its terminology to ensure the models complement each other. The model is complemented using, for each process domain, the key process areas discovered in each of the models is identified in **bold** in table VI. These key process areas ensure a good mapping to the source documents.

The CMMi process management has 5 key process areas a) organizational process focus; b) organizational process definition; c) organizational training; d) organizational process performance; and e) innovation and organizational deployment.

These areas can very well be used as-is in a software maintenance context. The content of the practices would be adapted to the specific software maintenance context but the key process areas would remain with the same names and meanings.

Decisions are required concerning the request management key process area. This is a new area and its components need to be defined. The CMMi had eight key process areas for project management. Since there is no project management in small software maintenance there is a need to look into the CMMi KPA's and see if some apply to software maintenance. The main differences are between request management and project management:

a. The planning, control and follow-up aspects could be used in tracking maintenance requests;
b. The contract, SLA's and supplier management appear in Niessik [*Nie02*], Camélia [*Cam94*] and Zitouni [*Zit95*] and can be useful to maintainers;
c. The project management concepts are not useful and will be eliminated from the model. Instead the problem management concepts are intriduced and used;
d. Last, the quantitative management KPA is kept to complement the control and follow-up of maintenance requests.

4 Process domains of software maintenance	Key Process Areas of Software Maintenance
Process Management	1- Maintenance Process Focus 2- Maintenance Process/Service definition 3- Maintenance Training 4- Maintenance Process Performance 5- Maintenance Innovation and deployment
Maintenance Request Management	1- Event Management 2- Maintenance Planning 3- Monitoring & Control of maintenance requests 4- SLA & Supplier Agreement Management 5- Quantitative Maintenance Management
Evolution Engineering	1- Transition 2- Operationnal Support 3- Evolution & Correction of software 4- Verification and Validation
Support to Evolution Engineering	1- Configuration Management 2- Process and Product Quality Assurance 3- Measurement, Decision and Causal Analysis 4- Reingineering/Reverse Engineering

Figure 9: DOMAINS AND KEY PROCESS AREAS OF SOFTWARE MAINTENANCE

Decisions are also required in the Evolution engineering process domain. The CMMi describes software development KPA's in this process domain: a) requirements definition; b) management of requirements c) technical solution; d) product integration; and e) verification

and validation. In this domain, the ISO12207 standard helps us to take the decision of using some of the developers processes for maintenance. In most areas the content of the CMMi practices will need to be adapted and simplified to reflect small maintenance engineering activities:

a. Requirements definition and management will be regrouped under one topic;
b. Product integration is kept with reduced content;
c. Technical Solution is kept with a reduced content;
d. Verification and validation moves from a support process to an engineering process in alignment with the CMMi;
e. Transition Management concepts are introduced as an engineering activity of software maintenance.

Finally some decisions are taken in the support domain. The CMMi had six KPA's: a) configuration management; b) quality assurance; c) analysis and measurement; d) organizational integration; e) problem resolution and decision analysis; and f) causal analysis. It is important to try and keep the alignment with the CMMi here as well. Reengineering, reverse engineering and reuse techniques to support maintenance engineering and used and defined.

6.2 The Model Construction

Once the architecture of the model was set the remainder of the ***SM^{mm}*** was built by integrating practices from the key reference documents relevant to software maintenance according to the following 9 steps:

Step 1 - Practices are taken from the mappings shown in annex A & B and the high level architecture (presented in figure 9);
Step 2 - A mapping is performed with the CMMi version 1.1 [*Sei02*]. CMMi practices may be slightly modified to accommodate this mapping and the particular use in a small enhancement context of software maintenance;
Step 3 - ISO 9001:2000 (ISO 90003:2004 interpretation guide [*Iso04*]). These practices are reviewed one by one, added and integrated to the model;
Step 4 - Other capability maturity model practices are then mapped to specific areas, depending on their coverage of the software maintenance domain (Cm^3-corrective maintenance of Kajko-Mattsson [*Kaj01a, Kaj01b*], IT service CMM [*Nie02*], the *Camélia* capability maturity model [*Cam94*] and finally the Zitouni model [*Zit95*];);
Step 5 - A reference is made to the IT infrastructure library individual recommendations for Service Delivery and Service Support [*Iti01a, Iti01b*];
Step 6 – A lighter mapping process is done with relevant portions of the Malcolm Baldrige examination criteria [*Mal04*];
Step 7 – Then the CobIT [*Cob00*] maintenance relevant practices are mapped where they apply to software maintenance activities;
Step 8 - References to relevant ISO/IEC and IEEE standards are added (ISO/IEC12207, ISO/IEC14764 and IEEE 1219);
Step 9 – Finally the author's experience and detailed recommendations are added to provide coverage of additional areas important to the software maintenance literature. These are based on professional benchmarks generated through the consensus of subject matter experts and validated in one peer-review process.

When practices are referenced/used from the CMMi, they go through the transformation process used by the Camélia project, if applicable:

1) Either removal of references to "development" or replacement of them by "maintenance" generalizes each practice;
2) References to "group" or to other specific organizational units are replaced by "organization";
3) Allusions to specific documents are replaced by examples pertinent to the maintainers.

The same types of transformations were applied when extracting practices from other standards or best practice guides. Assignment to a given level is based on the general guidelines of table IX. Furthermore:

- Practices considered fundamental to the successful conclusion of a maintenance practice are assigned to Level 2;
- Practices considered to be organization-wide in scope or fundamental to the continuous improvement of the software maintenance process are assigned to Level 3;
- Practices dealing with measurement or characterizing advanced process capability maturity (e.g. change management, integration of defect prevention, statistical process control and advanced metrics) are generally assigned to Level 4;
- Level 5 practices typically deal with advancing technology as it applies to process evolution, continuous improvement and strategic utilization of organization repositories.

6.3 The Resulting Model

The ***SM^{mm}*** is presented in Table VIII in more detail and includes 4 Process Domains, 18 KPAs, 74 Facets and 443 Practices. While some KPAs are unique to maintenance, some other were derived from the CMMi© and other models and modified slightly to map more closely to daily maintenance characteristics.

Table VIII

SM^{mm} Model Content

Process Domain	Key Process Area	Roadmap
Software Maintenance Process Management	Maintenance Process Focus	Responsibility and Communications ***Information gathering*** Findings Action plan
	Maintenance Process/Service Definition	Documentation and Standardization of processes/services Process/Service adaptation Communication processes /services Repository of processes/services
	Maintenance Training	Requirements, plans, and resources Personal training Initial training of newcomers Projects training on transition User training
	Maintenance Process Performance	Definition of maintenance measures Identification of baselines Quantitative management Prediction models
	Maintenance Innovation and Deployment	Research of innovations Analysis of improvement proposals Piloting selected improvement proposals Deployment of improvements Benefit measurement of improvements
Software Maintenance Request (MR) Management	Event and Service Request Management	Communications and contact structure Management of events and service requests
	Maintenance Planning	Maintenance Planning (1 to 3 yrs) Project transition planning Disaster Recovery planning Capacity planning Versions and upgrade planning Impact analysis
	Monitoring and Control of Service Requests and Events	Follow up on planned and approved activities Review and analyze progress Urgent changes and corrective measures
	SLAs and Supplier Agreements	Account Management of users Establish SLAs and contracts Execute services in SLAs and contracts Report, explain and bill services

Process Domain	Key Process Area	Roadmap
Software Evolution Engineering	Software Transition	Developer and owner involvement and communications
		Transition process surveillance and management
		Training and knowledge transfer surveillance
		Transition preparation
		Participation in system and acceptance tests
	Operational Support	Production software monitoring
		Support outside normal hours
		Business rules and functionality support
		Ad hoc requests/reports/services
	Software Evolution and Correction	Detailed design
		Construction (programming)
		Testing (unit, integration, regression)
		Documentation
	Software Verification and Validation	Reviews
		Acceptance tests
		Move to production
Support to Software Evolution Engineering	Software Configuration Management	Change Management
		Baseline configuration
		Reservation, follow-up, and control
	Process and Product Quality Assurance	Objective evaluation
		Identify and document non-conformances
		Communicate non-conformances
		Follow up on corrections/adjustments
	Measurement and Analysis of Maintenance	Define measurement program
		Collect and analyze measurement data
		Repository of maintenance measures
		Communicate measurement analysis
	Causal Analysis and Problem Resolution	Investigate defects and defaults
		Identify causes
		Analyze causes
		Propose solutions
	Software Rejuvenation, Migration, and Retirement	Redocumentation of software
		Restructuring of software
		Reverse engineering of software
		Reengineering of software
		Software migration
		Software retirement

6.4 The Model's Purpose

***SM**mm* 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.

The ***SM**mm* has been developed from a customer perspective, as experienced in a competitive, commercial environment. The ultimate objective of improvement programs initiated as a result of an ***SM**mm* assessment is increased customer (and shareholder) satisfaction, rather than rigid conformance to the standards referenced by this document.

A higher capability, in the ***SM**mm* context, means, that customer organizations are:

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

For a 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.

6.5 The Model Scope

Models are often an abstract representation of reality. For a better mapping with the maintainers' reality, the ***SM**mm* must include many of the essential perspectives of the software maintainer, and as much as possible of the maintainer's practical work context (see Figure 3).

These types of models are not intended to describe specific techniques or all the technologies used by maintainers. The decisions pertaining to the selection of certain techniques or technologies are specific to each organization.

Users of the model must instantiate the generic model in the context of their user organization. To achieve this, professional judgment is required to evaluate how an organization benchmarks against the generic model.

6.6 The *SMmm* Foundation

The ***SM**mm* is based on the Software Engineering Institute (SEI) Capability Maturity Model Integration for Software Engineering (CMMi©), version 1.1 [*Sei01*] and *Camélia* [*Cam94*]. The ***SM**mm* must be viewed as a complement to the CMMi©, especially for the processes that are common to both developers and maintainers, for example: a) process definition; b) development; c) testing; d) configuration management; and e) QA practices.

The architecture of the ***SM***^{mm} differs slightly from that of the CMMi© version 1.1. The most significant differences are:

1. A facet category to further define the Key Process Areas,
2. Detailed references to papers and examples on how to implement the practice.

The ***SM***^{mm} incorporates additional practices from the following topics:

1. Event and Service Request Management;
2. Maintenance Planning activities specific to maintainers (version, SLA, impact analysis);
3. Service Level Agreement;
4. Software Transition;
5. Operational Support;
6. Problem Resolution Process with a Help Desk;
7. Software Rejuvenation, Conversion and Retirement.

6.7 The ***SM***^{mm} Architecture

The CMMi© has recently adopted the continuous representation that has been successfully used in the past by other models such as: Bootstrap [*Boo91*], *Camélia* [*Cam94*] and ISO/IEC TR15504 (Spice) [*Iso98a*]. This model uses a continuous representation, as it helps to: a) conform to Spice recommendations; b) obtain a more granular rating for each facet and KPA; and c) identify a specific practice across maturity levels and identify its path from level zero (absent) to a higher level of capability maturity.

The ***SM***^{mm} is also based on the concept of a facet. A facet is a set of related practices which focuses on an organizational area or need, or a specific element within the software maintenance process. Each facet represents a significant capability for a software maintenance organization. Within a given facet, the level of a practice is based on its respective degree of capability maturity. The most fundamental practices are located at a lower level, whereas the most advanced ones are located at a higher level. An organization will mature through the facet. Lower-level practices must be implemented and sustained for higher-level practices to achieve maximum effectiveness. Each of the 6 capability maturity levels can be characterized, in the ***SM***^{mm} model, as follows (Figure 4):

Level	*Level Name*	*Risk*	*Interpretation*
0	*Incomplete*	*Highest*	*No sense of process*
1	*Performed*	*Very High*	*ad hoc maintenance process*
2	*Managed*	*High*	*basic request-based process*
3	*Established*	*Medium*	*state-of-the-art maintenance process*
4	*Predictable*	*Low*	*generally difficult to achieve now*
5	*Optimizing*	*Very Low*	*technologically challenging to attain*

Figure 10: *SM*^{mm} Capability Maturity Levels.

The capability maturity level definitions and the corresponding generic process attributes are described for each capability maturity level of the ***SM***^{mm} and presented in Table IX. [*Apr04*] describes how, over a two-year period, participating organizations contributed to the mapping of each relevant practice to a capability maturity level in the ***SM***^{mm} model.

Table IX
Process characteristics by process level

Level– Leve Name	Capability Level Definition	Process Generic Attributes
0- Incomplete Process	The process is not being executed by the organization, or there is no evidence that the process exists. Level 0 implies that the activity is not being performed by the organization	a) There is no evidence that the process exists; b) Upper management is not aware of the impact of not having this activity or process in the organization; c) The activity or process does not meet the goals stated by the model; d) There is no knowledge or understanding of the activity or process; e) Discussions concerning the activity or process take place, but no evidence can be found that the activity or process exists; f) Historical records show that the activity has been performed, but it is not being done at this time.
1- Performed Process	Improvised: Recognition that the practice is executed informally. Level 1 implies that something is being done or that the activity is close to the intention of the practice presented in the model. The execution of the practice depends on the knowledge and presence of key individuals. The practice is typically ad hoc and not documented. It is local and would not necessarily appear in another software maintenance group. There is no evidence that the attributes of the processes are systematically executed or that the activities are repeatable.	a) The organization is aware of the need to conduct this activity or process; b) An individual conducts the activity or process and the procedures are not documented (note: typically, staff must wait until this individual arrives on-site to learn more about the process; when this individual is not on-site, the activity or process cannot be executed fully); c) A few of the software maintainers execute this activity or process; d) Precise inputs and outputs of the activity or process cannot be recognized; e) There is no measure of the activity or process; f) The deliverables (outputs) are not used, not easily usable, and not kept up to date, and their impact is minimal; g) Who performs the activity or the qualifications/training required cannot be identified.
2- Managed Process	Awareness of the practice, which is deployed or a similar practice is performed. Level 2 implies that the practices suggested by the model are deployed through some of the software maintenance groups. What characterizes this level is the local and intuitive aspects of the activities or processes, which makes it difficult to harmonize them across all the software maintenance organizations.	a) The process is documented and followed locally; b) Training or support is provided locally; c) The goals of the process and activities are known; d) Inputs to the process are defined; e) Deliverables supporting the goals of the activity or process are produced; f) Qualitative measures of some attributes are performed; g) Individuals' names and qualifications are often described.

3- Established Process	The practice or process is understood and executed according to an organizationally deployed and documented procedure. Level 3 implies that the practice or process is defined and communicated, and that the employees have received proper training. Qualitative characteristics of the practice or process that are predictable are expected.	a) The practice or process suggested by the model is executed; b) The same practice is used across software maintenance groups; c) Basic measures have been defined and are collected, verified, and reported; d) Employees have the knowledge to execute the practice or process (i.e. implying that the roles and responsibilities of individuals are defined); e) The required resources have been assigned and managed to achieve the identified goals of the process; f) Techniques, templates, data repository, and infrastructures are available and used to support the process; g) The practice or process is always used by the employees; h) Key activities of the process are measured and controlled.
4-Predictabl Process	The practice is formally executed and quantitatively managed according to specified goals within established boundaries. There is an important distinction with respect to Level 4, in terms of the predictability of the results of a practice or process. The expression 'quantitatively managed' is used when a process or practice is controlled using a statistical control or similar technique well suited to controlling the execution of the process and its most important activities. The organization predicts the performance and controls the process.	a) Intermediate products of a process are formally reviewed; b) Conformance of the process has been assessed based on a documented procedure; c) Records of reviews and audits are kept and available; d) Open action items from reviews and audits are monitored until closure; e) Resources and infrastructures used by the process are planned, qualified, assigned, controlled, and managed; f) The process is independently reviewed or certified; g) Key activities of the process have historical data and an outcome that is measurable and controlled; h) Key activities have a numerical goal that is set and is attainable; i) Key activities have quantitative measures that are controlled in order to attain the goals; j) Deviations are analyzed to make decisions to adjust or correct the causes of the deviation.
5– Optimizing Process	The practice or process has quantified improvement goals and is continually improved. Level 5 implies continuous improvement. Quantitative improvement targets are established and reviewed to adapt to changes in the business objectives. These objectives are used as key criteria for improvements. Impacts of improvements are measured and assessed against the quantified improvement goals. Each key process of software maintenance has improvement targets.	a) Major improvements to process and practices can be reviewed; b) Innovations to technologies and processes are planned and have measurable targets; c) The organization is aware of and deploys the best practices of the industry; d) There are proactive activities for the identification activities of process weaknesses; e) A key objective of the organization is defect prevention; f) Advanced techniques and technologies are deployed and in use; g) Cost/benefit studies are carried out for all innovations and major improvements; h) Activities of reuse of human resources knowledge are performed; i) Causes of failure and defects (on overall activities/processes and technologies) are studied and eliminated.

7 EXAMPLES OF THE MODEL DETAILS

This section presents the details of two of the key process areas of the model.

7.1 The maintenance performance management process key area

Both the goals and target of the Performance Management Process, as well as the detailed practices are presented here.

7.1.1 Goals and objectives

Process performance management demands that the processes that have the greatest impact on the quality and process performance of the maintenance organization be identified first, followed by a definition of the measures and the establishment of a baseline (a set of references).

The goals of process performance management are:

- To document the rationale for the selection of the most important performance factors impacting processes and their activities, and which are key to achieving the quality targets;
- To measure and communicate the targets and quantitative results of the service levels to both customers and management.

The objectives of process performance management are:

- To identify the processes and the key activities of the software maintenance organization that impact performance, and which are to be used for analysis;
- To establish a baseline consisting of the key software maintenance processes and activities;
- To identify and establish the measures of the performance of the selected processes/activities;
- To establish models for predicting the performance of the software maintenance processes/activities.

This performance management process is not complete in and of itself. It requires practices referenced in other process areas of the **SM-*MM*** model for it to be fully operational. The first link is to the facet *Quantitative Management* – (see table VIII), which provides more information on how to use a reference point (baseline) of the performance of processes and of its models. The second link is to *Measurement and Analysis of Maintenance* – (see table VIII), to obtain more information on how to specify a measure, collect the data and conduct an analysis of the data.

Once the maintenance performance management process has been successfully implemented, it will be observed that:

- The measures of quality and of process performance have been established for the production software, intermediary products and software processes;
- The measures are harmonized and relate to the normalized processes;
- The data collection activities are performed at the operational level and the data are stored in the corporate repository;
- The baseline has been created, validated and documented.

7.1.2 Detailed practices

Only detailed practices of capability maturity levels from 0 to 3 are presented next.

7.1.3 Level 0 and 1 practices

The individual practices are assigned to one of five levels of capability maturity. At level 0, there is only one practice;

Pro0.1 The software maintenance organization does not conduct process performance measurement on its processes.

These organizations just perform the daily work of software maintenance.

At level 1, there are two practices that are presented below:

Pro1.1 Individuals and employees interested in this domain implement Individual initiatives of process or product measurement.

In these software maintenance organizations, the software maintenance managers and programmers develop process and product measurements of their own initiative. These definitions are personal and are rarely: a) shared with other organizations; or b) used to better manage or improve either process or product. They are used typically to explain, internally or to a user, an individual event or situation that occurred.

Pro1.2 Some qualitative process and product measures are collected by the software maintenance organization.

The employees of the software maintenance organization have established relationships with their end user counterparts and obtained qualitative performance measures on how they are performing. These comments and observations typically appear in conversations and e-mails.

7.1.4 Level 2 practices

At this capability maturity level, process performance addresses basic considerations and typically differs across the various units which conduct software maintenance in the organization. Qualitative information is normally collected by the manager and his employees and reported in weekly and monthly meetings. When quality and performance targets are established, this information is used for local, short-term improvement and is based on individual priorities.

Pro2.1: Some processes and key products of software maintenance have identified measures.

At this capability maturity level, the software maintenance organization should have identified a basic set of process measures. These measures are collected and are typically used in: a) the

weekly management meeting; and b) communications with customers. Typical measures found at this level are based on what could be called measures for the management of queues [*Abr01*], which means that process performance has not yet been addressed fully or completely understood. These measures are for example:

- The number of outstanding requests;
- The average waiting time before being serviced;
- The estimated number of days in the queue;
- The number of requests completed;
- The number of requests closed for this period, opened during this period and still pending;
- A comparison of estimates versus actual costs.

Pro2.2 Some quality and performance targets exist in the software maintenance organization.

At this level of capability maturity the software maintenance organization must identify basic quality and performance targets. Setting quality targets requires that the current performance is looked at and that realistic goals are established. These targets are typically used in the weekly management meeting and in the communications with customers. For example:

a) the availability percentage of all software under their responsibility , i.e. 98%;
b) the degree (%) of satisfaction of customers based on surveys;
c) the limit of available overtime hours of the maintenance staff; and
d) the average waiting time for a change request, by category, to be completed.

Pro2.3 The reference points (baselines), for the current measures are stored, used and reviewed with the various stakeholders (customers, sponsors, program managers and maintenance employees) for review, discussion and improvement.

At this capability maturity level, the current level of performance is measured and this data is captured, stored and communicated to the many stakeholders. The objective is to establish and communicate the current level of performance of the maintenance activities and production software to all concerned. At this capability maturity level, this data is typically accumulated in a local, and sometimes personal, repository and is typically used by the software maintenance manager to explain and analyse specific situations and events. The objective of this practice is the sharing and acceptance of an agreed-upon, common and available reference (baseline), which describe the current performance levels of the maintainer.

7.1.5 Level 3 Practices

At this capability maturity level, the definition of the measures becomes more precise, and available, and is now standardized among all software maintenance units of the organization. The process performance definition activity must be considered as a process improvement activity to ensure alignment with the overall quality objectives of the organization. The key activities of the standardized software maintenance processes are identified as candidates to be controlled and measured. Quality and performance attributes of operational software are also defined. Measure targets and reference points (baselines) are established and maintained, and a rationale describes its variation within established upper and lower boundaries. The definition of

each measure is documented, and the data collection and data validation activities required are identified. Customers should perceive a harmonization and openness of the maintainer's activities, services and measures. The software maintenance personnel are trained in process performance measurement activities. The measurement data is collected, validated and integrated into the corporate measurement repository. Maintenance personnel collect measurement data, daily, as part of their operational activities.

Pro3.1 The software maintenance measurement programme is defined and treated as a project. Special attention is given to risk management in order to minimize the risk of failure of this programme.

To achieve this best practice, a measurement programme must be set up (with multiple controlled implementation steps or stages) and its risk managed. It is especially important to manage the measure definition, collection and verification activities, as is done at the operational level. Risk management can be divided into three steps: 1) definition of a risk management strategy; 2) identification and analysis of the risks; and 3) a check of the identified risks, including the implementation of mitigation, if necessary [*Sei02, PA148.N104*]. The measurement can be also integrated with other initiatives (operations or development).

Pro3.2 The software maintenance organization identifies its key processes and their activities, and defines quality and performance measures.

This best practice requires that key activities of the software maintenance processes be identified and investigated to understand which ones contribute the most to quality and performance. [*Sei02, PA164.IG101.SP101*]. The maintenance process has been deployed across software maintenance business units. At this capability maturity level, the measurement activity must be coordinated across all software maintenance units of the organization. It is also necessary to align this activity with other software engineering measurement initiatives to ensure a cohesive approach for the whole Software Engineering organization. Ideally, all the measurement data should be integrated in one repository. Harmonization of measures in an organization is key [*Sei02, PA164.IG101.SP102.SubP103*]. The definition of the measures that will be part of the performance analyses of software maintenance must be established and maintained [*Sei02, PA164.IG101.SP102*]. The process measures are determined according to their perceived value for the maintenance organization and their customers. These measures should cover the whole software life cycle, including maintenance [*Iso04, 6.4.2, Cam94, 3.4.3*]. Measures on which standardized techniques or processes can be used are defined and set up in accordance with a formal procedure [*Cam94, 1.5.3.5*]. An organization–wide, approved and funded measurement programme supports the measurement activities and the analysis of these measures [*Iso04, 6.4*] [*Cam94, 3.4.3.1*]. The standardized software maintenance processes are used to select where data are to be collected and what analysis is required [*Cam94, 3.4.3.2*].

Pro3.3 The software maintenance organization identifies its key products and production software, and defines their quality and performance measures.

'It is not practically possible to measure all sub-characteristics internally or externally for all parts of a large software product. Similarly it is not usually practical to measure quality in use for all possible user-task scenarios. Resources for evaluation need to be allocated between the

different types of measurement dependent on the business objectives and the nature of the product and maintenance processes.' [*Iso04*] The maintainer must identify which perspective of the products to measure, i.e. those that have a significant impact on the customer and on the quality of the operational software. To achieve this best practice, the maintainer must identify key product measures and document why measuring them is important. In order to do this, all the software maintenance departments must share the same measures, so that an integrated approach to service level management and reporting will begin to emerge.

The intermediary products (for example: technical documentation, software testing activities) are rarely presented to the customer, but are key to software quality. The concept of internal versus external software measures (see Figure 10) is introduced here. Achievement of software product quality is based on the execution of the operational and supporting processes in software product quality that can be measured internally (typically, by static measures on the code) or externally (typically, by measuring the behavior of the source code when executed).

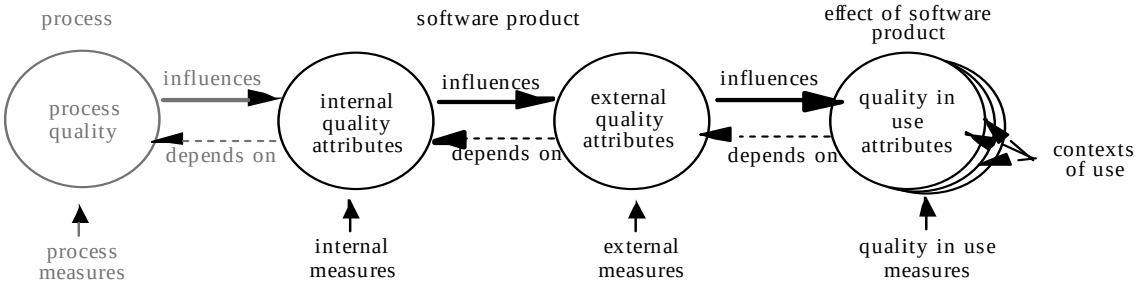

Figure 11: ISO/IEC 9126 - Model of quality in the product life cycle

The objective is for the product to have the required effect in a particular context of use [*Iso04*]. The external measures are used to reflect the attributes of the quality of the software that are apparent and important to customers. For example, availability measurement of the software is done, as this is important to the customer. Other examples of such important external measures are:
a) measures of, and follow-up on, delivery times of a specific request [*Cam94, 3.4.3.5*];
b) measures of software failure, which are often associated with the identification of which software unit was involved in the failure [*Cam94, 3.4.3.7*].

Pro3.4 The software maintenance organization's key processes, products and operational software have quality and process performance targets.

In this best practice, the maintainer must set quality and performance targets, and these must be established and maintained [*Sei02, PA164.IG101.SP103*]. The following attributes are necessary [*Sei02, PA164.IG101.SP103.N101*]:

- Establishment of targets that take into account the business objectives;
- Establishment of targets that take into account previous performance of the activities and operational software;
- Definition of productivity and process execution aspects in order to judge quality;
- Identification of the process limits (by inherent variability or natural limits) on key processes or key maintenance products.

Pro3.5 A measure of a key process, product or operational software for which the software maintenance organization services has a validated reference point (baseline) which is used in analysis, control and improvement follow-up.

The reference points (baselines) of the software maintenance measures must be documented and include some level of detail [*Sei02, PA164.IG101.SP104.N101*]. For example, it will be necessary to identify:

- The key activity of the process targeted by the measure;
- The sequence of measurement activities;
- How representative of the software maintenance work this measure is.

It is possible to have more than one reference point for the maintenance organization. These different values can represent the performance of different organizational units of the maintainer when they execute a process [*Sei02, PA164.IG101.SP104.N102*]. A number of measures, covering maintainability and other key characteristics of products, are collected and used to manage and improve the maintenance processes [*Iso04, 6.4.1, Cam94, 3.4.3.4*]. Slight adaptations of a normalized process can lead to a reduction in the possibility for comparison of the various organizational units. Some explanation of the adaptations and of the reference point values must be documented to allow for such comparisons [*Sei02, PA164.IG101.SP104.N103*].

Pro3.6 Models of the software maintenance process performance are established.

Prediction models for software maintenance activities must be developed to achieve this best practice. These models are typically used to estimate, or predict, the results of maintenance processes based on historical and current data. An attempt is made, therefore, to predict the future behaviour of the processes based on current data [*Sei02, PA164.IG101.SP105. N101*]. For example [*Sei02, PA164.IG101. SP105.N102*]: The maintenance organization will be able to use these models to estimate and predict the delivery of services, and of intermediary products and software product versions. The organization can also attempt to evaluate the return on investment of the process improvement activities. Maintenance employees can use the models to estimate, analyse and forecast the effort and cycle times of different maintenance activities. It will also be possible to attempt to forecast the overtime of employees and the availability of software. In addition, the models can be used to help size a modification and to predict failure rates based on the size of the modification [*Abr95*]. However, it is important to collect enough data to ensure the statistical validity of the analyses.

The use of the models will grow as their prediction capabilities improve and as the data collected is managed. It will be possible, for example, to use them in situations like the following [*Sei02, PA164.IG101.SP105.N102*]: Analysis of process or product data using a formal procedure [*Cam94 3.4.3.9, Iso04*]. One should not, however, lose sight of the fact that the models are best used with the key maintenance activities and products, which have a visible impact with customers and stakeholders [*Sei02, PA164.IG101.SP105. N103*].

Pro3.7 Models of software performance are established.

Performance engineering models profiling the operational software under the maintainer's responsibility must be developed and used to achieve this best practice. These models are used

to estimate, or predict, the operational performance of software using historical data. An attempt will be made, therefore, to use these models to predict future behaviour using current data [*Sei02, PA164.IG101.SP105.N101*].

More information can be found in the Performance Engineering maturity model [*Sch99*] in which it is recommended that the following activities be considered at capability maturity level 3:

- The Performance Engineering process should be taken into account throughout the entire software process, and all available Performance Engineering methods and tools be used comprehensively with regard to existing performance risk.
- Performance-relevant product and resource metrics should be selected for Performance Engineering use and standardized within the organization and these metrics be stored and managed in appropriate database systems to guarantee a continuous flow of experiences.
- The performance requirements of the customer, which are defined in the system analysis phase, should be used as success criteria in the final inspection test. Furthermore, they should be arranged in service level agreements (SLA) with the provider of the information system.
- An initial organizational structure for the entire PE process should be defined and introduced step-by-step in level 3.

7.2 The Management of Service requests and events key process area

At the detailed level for each KPA, maintenance goals and key practices have been identified based on the literature on software maintenance. This section of the article presents, as an example, a detailed description of one of the 18 KPA of the ***SM***^{mm}: 'Management of Service Requests and Events'.

7.2.1 Overview

The management of service requests and events for a software maintainer combines a number of important service-related processes.

These processes ensure that events, reported failures or modification requests and operational support requests are identified, classified, prioritised and routed to ensure that the SLA is fully met.

An event, if not identified and managed quickly, could prevent service level targets from being met and lead to user complaints about: a) the slowness in processing of a specific request; or b) unmet quality targets for an operational software (ex: availability or response time).

7.2.2 Objectives and goals

This KPA covers the requirement that users are made aware of the maintenance workload and authorize and agree on maintenance priorities. Maintainers must also oversee software and operational infrastructures as well as production software behavior (availability, performance, reliability, stability as well as the status of the software and its infrastructure). When priorities

change, maintainers must ensure that the maintenance workload will be reassigned quickly, if necessary.

The goals of Management of Service Requests and Events are as follows:

- To proactively collect, and register all requests for services (customer-related, or internally generated);
- To oversee the behavior of the software and its infrastructures during the last 24 hours, to identify events that could lead to missing SLA targets;
- To develop a consensus on the priorities of service requests (in the queue or being processed);
- To ensure that maintainers are working on the right (and agreed-upon) user priorities;
- To be flexible and have the ability to interrupt the work in progress based on new events or changed priorities;
- To proactively communicate the status of the service, planned resolution times, and current workload.

To ensure that the agreed-upon service levels are met, the objectives of Management of Service Requests and Events are:

- to ensure that events and service requests are identified and registered daily;
- to determine the relative importance, within the current workload, of new events and service requests; and
- to ensure that the workload is focused on approved priorities.

The maintainer must also communicate proactively about failures, and unavailability of software (including its planned preventive maintenance activities).

Like performance management process, this process is not fully complete in and of itself. It requires practices referenced in other process areas of the **SM-*MM*** model for it to be fully operational. As an example, linkages are required to: *Impact Analysis, Service level Agreement, Operational Support and Causal Analysis & Problem Resolution.*

Once the Management of Service requests and events process has been successfully implemented, it will be observed that:

- Maintenance work is centered on user priorities and SLAs;
- Interruptions of maintenance work are justified, and are authorized by users and SLAs;
- The maintenance organization meets its agreed- upon levels of services;
- Proactive operational software surveillance ensures rapid preventive action;
- Status reports, on failures and unavailability, are broadcast quickly and as often as required until service restoration.

7.2.3 Detailed practices

The individual practices are assigned to one of five levels of capability maturity. Examples of detailed practices follow for capability maturity levels 0 to 3.

7.2.4 Level 0 and 1 practices

At level 0, there is only one practice:

Req1.0.1 The software maintenance organization does not manage user requests or software events.

Maintenance organizations operating at this capability maturity level perform the daily work of software maintenance without being formally accountable for their activities and priorities to the user community. At level 1, two practices are documented in the model:

Req1.1.1 Request and event management is managed informally.

Req1.1.2 An individual approach to managing user requests and events is based mainly on personal relationships between a maintainer and a user.

The software maintenance organizations, which operate at this capability maturity level have typically had informal contacts with some users and none with others. Records of requests or events are not standardized. Service is given unevenly, reactively and based on individual initiatives, knowledge and contacts. The maintenance service and workload are: a) not measured and, b) not based on user priorities; and c) seldom publicized or shared with user organizations.

7.2.5 Level 2 practices

At level 2, the service requests are processed through a single point of contact. Requests are registered, categorized and prioritised. Approved software modifications are scheduled to a future release (or version). Some local effort of data collection emerges and can be used to document maintenance costs and activities through a simple internal accounting procedure.

Req1.2.1: There is a unique point of contact to provide direct assistance to users.

At this capability maturity level, the software maintenance organization should have identified a point of contact for each software service request, software and user**.**

Req1.2.2 A Problem Report (PR) or Modification request (MR) is registered and used as a work order (also sometimes called a ticket**) by the maintainer.**

At level 2, the software maintenance organization maintains records of each request, and uses them to manage the incoming workload.

Req1.2.3: Every request and event is analyzed, categorized, prioritized, and assigned an initial effort estimate.

Maintainers classify the service requests and events according to standardized categories. Each request is assessed to determine the effort required. Pfleeger [*Pfl01*] adds that an impact analysis is carried out, and, in each case, a decision as to how much of the standard maintenance process will be followed based on the urgency and costs that can be billed to the customer.

Req1.2.4: Approved modifications are assigned, tentatively, to a planned release (version) of a software application.

Maintainers are starting to regroup changes and plan for releases and versions. Each request is allocated to a planned release**.**

Req1.2.5: The service level measurement reports are used for invoicing maintenance services.

At level 2, the maintainer uses the same processes and service-level reports for invoicing maintenance services and budget justification.

Req1.2.6: A summary of maintenance cost data is presented. The invoice is based on a limited number of key cost elements, those most important to the maintainer.

The maintainer must be in a position to report on all the service requests worked on during a reporting period (e.g. monthly). ISO/IEC 14764, states that analyzing completed maintenance work, by maintenance categories, helps in gaining a better understanding of maintenance costs.

7.2.6 Level 3 practices

For the sake of brevity, only the level 3 list of practices is presented here:

Req1.3.1: Various alternatives are available to users to obtain help concerning their software applications and related services.

Req1.3.2: Users are kept up to date on the status of requests and events.

Req1.3.3: Proactive communications are established for reporting failures, as well as for planned preventive maintenance activities which impact the user community.

Req1.3.4: A decision-making process is implemented to take action on a maintenance service request (e.g. acceptance, further analysis required, discard it).

Req1.3.5: Failures and user requests, including modification requests, are registered (tickets) and tracked in a repository of maintenance requests, in conformity with written and published procedures.

Req1.3.6: Procedures on the registration, routing, and the closing of requests (tickets) in the repository of maintenance requests, are published and updated.

Req1.3.7: The mandatory and optional data fields on the user request form are standardized.

Req1.3.8: Problem Reports (PR) document includes detailed data related to reported failures.

Req1.3.9: The request and event management process is linked to the maintenance improvement process.

Req1.3.10: Standardized management reports documenting requests and events are developed and made available to all Software Engineering support groups and to users.

Req1.3.11: A process is implemented to decrease the waiting time of requests in the service queue.

Req1.3.12: Data on actual versus planned maintenance costs are documented, as well as details on the usage and the costs for all maintenance services (e.g. corrective, perfective, adaptive …);

Req1.3.13: The invoice includes the detailed costs of all services, by software application.

8 CONTRIBUTIONS OF THIS RESEARCH

Contribution to knowledge, in this study, is closely related to the research questions and problems of software maintenance. This research applies the concept of capability maturity model to the of software maintenance function. Based on the results:

1) It presents a comprehensive inventory of the maintenance processes and activities;
2) It identifies that the current representation of software maintenance in international standards (ISO12207 and ISO14764) covers only partially the processes and activities of software maintenance;
3) It confirms that there is no maturity model proposal that covers the entire set of processes and activities of software maintenance; and
4) It proposes a more comprehensive model of the software maintenance function.

This research has also contributed to the enhancement of the SWEBOK chapter dedicated to software maintenance. Many of the findings of this study have beeen submitted within the Ironman review cycle of the SWEBOK project, and have been accepted by the SWEBOK associate editor (Thomas Pigosky) for the maintenance chapter. In addition, this PhD candidate has been nominated as the co-editor of the maintenance chapter for the SWEBOK Ironman version.

The findings of this research are also used to supply the editorial team with comments through the Canadian Standards Association for consideration in the next version of the ISO14764 International standard on software maintenance. A French book describing the whole model has been submitted to an editor in July 2004 [*Apr04b*].

Some of the key findings of this research have been progressively made public during 2003 and 2004 at software engineering conferences.

Following are publications that are related to this research :

A.April, A.Abran, R.Dumke, "SM^{cmm} to Evaluate and Improve the Quality of Software Maintenance Process: Overview of the model," SPICE 2004 Conference on Process Assessment and Improvement, Critical Software SA, The Spice User Group, Lisbon (Portugal), Apr. 27-99, 2004, pp. 19:32.

A.April, A.Abran, R.Dumke, (2004) "Assessment of Software Maintenance Capability: A model and its Design Process", IASTED 2004 Conference on Software Engineering, Innsbruck (Austria), Feb. 16-19.

A. April, A. Abran, R. Dumke, “What you need to know about Software Maintenance”, Maintenance Journal, Feb. 2004 (http://www.lrgl.uqam.ca/team/showmem.jsp?author=april&cv_name=april.html)

A.April, A.Abran, R.Dumke, (2004) "Assessment of Software Maintenance Capability: A model and its Architecture", CSMR 2004, 8th European conference on Software Maintenance and Reengineering, Tampere (Finland), Mar. 24-26.

A. April, A. Abran, R. Dumke, "Software Maintenance Capability Maturity Model (SM-CMM): Process Performance Measurement", 13th International Workshop on Software Measurement – IWSM 2003, Montréal (Canada), Springer-Verlag, Sept. 23-25, 2003, pp. 311-326.

April A, Abran A, Bourque P. (2003) Analysis of the knowledge content and classification in the SWEBOK chapter: Software maintenance. Technical Report 03-001 of the ETS Software Engineering Laboratory, 12 pp. [On line] http://www.lrgl.uqam.ca/team/showmem.jsp?author=april&cv_name=april.html (link tested on July 2nd 2004)

A.April, D.Al-Shurougi, (2002), "Software Maintenance Productivity", ICB/ASAY "The Role of Quality Maintenance in Cost Minimisation Conference, Bahrain, May 27-28.

A. April, J. Bouman, A. Abran, D. Al-Shurougi, (2001) Software Maintenance in a Service Level Agreement: European Software Measurement Conference, Heidelberg, Germany, May 8-11.

April, A., Al-Shurougi, D. (2000). Software Product Measurement for supplier Evaluation, FESMA-AEMES Software Measurement Conference, Madrid (Spain), October 18-20, 2000, http://www.lrgl.uqam.ca/publications/pdf/583.pdf (link tested May 11, 2004).

Apart from this preprint:
- A second journal article has been submitted to the Journal of Software Maintenance and Evolution;
- A paper entitled Software Maintenance Productivity measurement: how to assess the readiness of your organization was submitted to IWSM2004.

Finally since the publication of those papers many organizations (Brazil, France, USA, Germany and Canada) and researchers (China and India) have enquired about the model and shown interest in its detailed findings.

9 SUMMARY

This article has presented a maintenance-specific capability maturity model: The Software Maintenance Maturity Model – (***SM***^{mm}).

It has presented a literature review of the software maintenance and software engineering maturity models followed by an identification of the maintainer's unique processes and activities. It also described the approach taken to build the model, as well as the model purpose, scope and foundation. Examples of the content of two key process areas are also presented. This ***SM***^{mm} model is based on the architecture of the model developed by the SEI of the Carnegie Mellon University of Pittsburgh to evaluate and improve the process of software development.

This article has investigated the following research questions:

1) The list of processes and activities as well as the unique processes and activities of software maintenance;

2) Software maintenance refers to software development processes and activity but must adapt them to its specificity;

3) That the unique processes of software maintenance are not well reflected in the current international standards;

4) There is not currently a maintenance-specific capability maturity model that covers the entire set of software maintenance specific processes and activities;

5) It proposed architecture of a maintenance-specific capability maturity model that could address the entire set of software maintenance unique activities as well as presented examples of detailed practices for two key process areas.

The motivation for this ***SM***^{mm} model was to contribute to addressing the quality issues of the maintenance function and to suggest further directions for improvements. Empirical studies on the use of the ***SM***^{mm} as a tool for continuous improvements in maintenance management could contribute to developing a better understanding of the problems of the software maintenance function.

Further field studies are required to experiment and validate this software maintenance improvement model. This will ensure that the key practices suggested by maintenance experts or described in the literature are positioned at the correct level of maturity within this maintenance assessment model.

REFERENCES[5]

[Abr91] A.Abran, H. Nguyenkim, (1991) Analysis of Maintenance Work Categories Through Measurement. Proceeding of the Conference on Software Maintenance IEEE, Sorrento, Italy, October 15-17, pp. 104-113.

[Abr93] A.Abran, H. Nguyenkim, (1993), "Measurement of the Maintenance Process from a Demand-based Perspective", Journal of Software Maintenance: Research and Practice, 5 (2), 63-90.

[Abr93a] A.Abran, (1993) Maintenance Productivity & Quality Studies: Industry Feedback on Benchmarking. Proceedings of the Software Maintenance Conference, ICSM93, Montréal September 27-30, Invited paper pp. 370.

[Abr95] A.Abran, M. Maya, (1995) "A Sizing Measure for Adaptive Maintenance Work Products", International Conference on Software Maintenance - ICSM-95, Opio (France), IEEE Computer Society Press, Los Alamitos, CA.

[Abr01] A.Abran and J.W. Moore Executive Editors; P. Bourque and R. Dupuis: Editors; L.L. Tripp Chair of IAB, (2001) Guide to the Software Engineering Body of Knowledge - Trial Version, IEEE Computer Society Press, Dec.

[Abr04] A.Abran, J.W. Moore (Exec. Eds), P. Bourque, R. Dupuis (Eds), (2004) *Guide to the Software Engineering Body of Knowledge – 2004 Version*, IEEE Computer Society, Los Alamos. [On line] www.swebok.org (link tested on July 2nd 2004)

[Apr01] A.April, J.Bouman, A.Abran, D. Al-Shurougi, (2001), "Software Maintenance in a Service Level Agreement: Controlling the Customer Expectations", Fourth European Software Measurement Conference, FESMA, Heidleberg, Germany, May.

[Apr02] A.April, D.Al-Shurougi, (2002), "Software Maintenance Productivity", ICB/ASAY "The Role of Quality Maintenance in Cost Minimisation Conference, Bahrain, May 27-28.

[Apr02a] A.April (2002), Revue Critique de la littérature dans le cadre du doctorat en génie à l'ÉTS, Modèle de référence pour l'évaluation de la capacité d'entretien du logiciel, La maintenance du logiciel, 10 Novembre 2002, 72p.

[Apr02b] A.April (2002), Revue Critique de la littérature Partie 2, Les modèles de référence pour l'évolution des processus du logiciel, Isa Town, Bahrain, 30 Décembre 2002, 94p.

[Apr03] April A, Abran A, Bourque P. (2003) Analysis of the knowledge content and classification in the SWEBOK chapter: Software maintenance. Technical Report 03-001 of the ETS Software Engineering Laboratory, 12 pp. [On line] http://www.lrgl.uqam.ca/team/showmem.jsp?author=april&cv_name=april.html (link tested on July 2nd 2004)

[Apr03a] April, A. (2003) Problématique de Recherche : Modèle d'évaluation de la capacité à Maintenir le Logiciel, École de Technologie Supérieure, DGA1010, 19 Janvier 2003, 48p.

[Apr04] A.April, A.Abran, R.Dumke, (2004) "Assessment of Software Maintenance Capability: A model and its Design Process", IASTED 2004 Conference on Software Engineering, Innsbruck (Austria), Feb. 16-19.

[Apr04a] A.April, A.Abran, R.Dumke, (2004) "Assessment of Software Maintenance Capability: A model and its Architecture", CSMR 2004, 8th European conference on Software Maintenance and Reengineering, Tampere (Finland), Mar. 24-26.

[Apr04b] A.April, A.Abran (2004) Modèle d'Évaluation de la Capacité à Maintenir le Logiciel, L'amélioration de la maintenance du logiciel – Théorie et Pratique, Première Édition, soumis à l'éditeur LozeDion, Juillet 2004.

[Aps01] Australian Public Service Commission. (2001) *The Human Resource Capability Model*, [On line] http://www.apsc.gov.au/publications01/hr360.htm (Link tested on July 2nd 2004)

[Art88] Arthur, L. (1988). Software Evolution: The Software Maintenance Challenge, John Wiley & Sons.

[Baj98] Bajers, F. (1998), "How to introduce maturity in software change management", Technical Report R98-5012, Department of Computer Science, Aalborg University, Denmark.

[5] Internet adresses are valid at time of publication.

[Ban93] Banker RD, Datar SM, Kemerer CF, Zweig D. (1993) Software complexity and maintenance costs. *Communications of the ACM*; **36**(11):81–94.

[Bar95] Barghouti, N., Rosenblum, D., et al. (1995). Two case studies in modeling real, corporate processes, Software Process: Improvement and Practice, 1(1), 17-32.

[Bas96] V.Basili, L.Briand, S.Condon, Y.Kim, W.Melo, J.Valett, (1996) Understanding and Predicting the process software maintenance releases. Proceedings of the International Conference on Software Engineering, IEEE.

[Ben00] Bennett, K.H. (2000) Software Maintenance: A Tutorial. In Software Engineering, edited by Dorfman and Thayer. IEEE Computer Society Press: Los Alamitos, CA; 289-303 pp.

[But95] Butler, K. (1995). The Economic Benefits of Software Process Improvement, Crosstalk, 8(7), July issue, 14-17.

[Boe87] Boehm, B.W. (1987). Industrial software metrics top 10 list, IEEE Software, 4(5), September issue, 84-85.

[Boo91] Bootstrap, (1991). Esprit project #5441, European Commission, Brussels, Belgium.

[Bou96a] P.Bourque, (1996) An Overview of Software Maintenance research at SEMRL-UQAM. International Software Benchmarking: Engineering and Measurement Issues. Montreal: Université du Québec Montréal. [On line] http://www.lrgl.uqam.ca/team/showmem.jsp?author=Bourque&cv_name=pierre.html (Link tested on July 2nd 2004)

[Btu90] Telstar, (1990) Software Development Methodology, Pocket Reference Guide, British Telecommunications, UK, Release II.1, 61 pp.

[Bur96] Burnstein, I., Suwannasart, T., Carlson, C. (1996), "Developing a Testing Maturity Model: Part I", Crosstalk Journal, August, 21-24 [On line]. http://www.stsc.hill.af.mil/crosstalk/. (link tested on 11 November 2002).

[Bur96a] Burnstein, I., Suwannasart, T., Carlson, C., (1996), Developing a Testing Maturity Model: Part II", Crosstalk Journal, September, [On line]. http://www.improveqs.nl/pdf/Crosstalk%20TMM%20part%202.pdf (link tested on 11 October 2003).

[Cam94] *Camélia*. (1994), "Modèle d'évolution des processus de développement, maintenance et d'exploitation de produits informatiques", Projet France-Quebec, Version 0.5, Montréal (Canada).

[Car92] Cardow, J. You Can't Teach Software Maintenance! Proceedings of the Sixth Annual Meeting and Conference of the Software Management Association 1992.

[Car94] D.Carey, (1994), "Executive round-table on business issues in outsourcing - Making the decision", CIO Canada, June/July.

[Cfi02] Voice of the Customer Surveys, Claes Formell International, [On line] http://www.cfigroup.com/home.htm (link tested on July 2nd 2004)

[Cob00] IT Governance Institute. (2000) CobiT, Governance, Control and Audit for Information and Related Technology. ISACA, Rolling Meadows, Illinois: 81 pp.

[Col87] M.Colter, (1987) The business of Software Maintenance, Proceedings 1st workshop Software Maintenance, Computer Sccience Department, University of Durham, South Road, Durham, DH1 3LE, UK,.

[Cra02] Crawford, J.K. (2002), "Project management Maturity Model, Providing a proven path to project management excellence", Marcel Dekker/Center for business practices.

[Des99] Desautels, L.D. (1999). Financial Management Capability Model, Auditor general of Canada.

[Dek92] S.M. Dekleva. (1992), "Delphi Study of Software Maintenance Problems", ICSM - International Conference on Software Maintenance, IEEE Computer Society Press, 10-17.

[Dor02] M. Dorfman and R. H. Thayer. (2002) *Software Engineering, 2nd Edition. Volume 1 – The Development Process*. Edited by Richard H. Thayer and Merlin Dorfman, IEEE Computer Society Press, ISBN 076951555X.

[Dor97] M. Dorfman and R. H. Thayer. (2002) *Software Engineering, 2nd Edition. Volume 2 – The Supporting Processes*. Edited by Richard H. Thayer and Mark Christensen, IEEE Computer Society Press, ISBN 0769515576.

[Dov96] Dove, R., Hartman, S., Benson, S. (1996), "A Change Proficiency Maturity Model, An Agile Enterprise Reference Model with a Case Study of Remmele Engineering", Agility Forum, AR96-04, December.

[Fos87] Foster, J.R., Munro, M. (1987) A Documentation Method Based on Cross-Referencing, Proceedings of the IEEE Conference on Software Maintenance, IEEE Computer Society Press, Los Alamitos, California, pp 181-185.

[For92] Fornell, G.E. (1992) cover letter to report, "Process for Acquiring Software Architecture," July 10, http://www.stsc.hill.af.mil/resources/ tech_docs/gsam2/chap_2.DOC (link tested May 11th 2004).

[Fug96] Fuggetta, A. Wolf, A. Software Process, John Wiley & Sons, New York, 1996, 160 pp.

[Gil94]

[Gil99] Gillies, A., (1999). A model for measuring the maturity of information and IT in primary care in the UK, University of Central Lancashire, CHERA conference.

[Gla92] Glass, R.L. (1992) Building Quality Software, Prentice Hall, Englewood Cliffs, New Jersey.

[Gla81] Glass, R.L., Noiseux, R. (1981) Software Maintenence Guidebook, prentice-Hall.

[Gra87] R. Grady, D. Caswell, (1987) Software Metrics:Establising a Company wide Program. EnglewoodCliffs, NJ, Prentice-Hall.

[Han93] Hanna, M. (1993) Maintenance Burden Begging for a Remedy, Datamation, April, pp.53-63.

[Hal87] Hall, P.V.A. (1987) Software components and reuse: Getting more out of your code. *Information Software Technology*, Vol. 29, No. 1, Feb, pp. 38-43.

[Hp90] Corporate Engineering, (1990) Software Quality and Productivity Guide – Best Practice in Software Development at Hewlett-Packard.

[Hum00] Humphrey, W. (2000). Managing Technical People – Innovation, teamwork and the software Process, Addison-Wesley.

[Iee90] IEEE Std 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology.

[Iee98] IEEE Std 1219, (1998). Standard for Software Maintenance, IEEE Computer Society Press.

[Iso95] ISO/IEC 12207, (1995). Information Technology – Software Life Cycle Processes, ISO and IEC, Geneva.

[IFP94] Guidelines to Software Measurement, version 1.0, International Funtion Point User Group, 1994.

[Isb04] International Benchmarking Software group (2003). Data Collection Questionnaire Application Software Maintenance and Support, Version 1.0, [On Line] http://www.isbsg.org.au/html/index2.html (link tested on June 25th 2004).

[ISO98] ISO/IEC 14764, (1998). Software Engineering-Software Maintenance, ISO and IEC, Geneva.

[ISO98a] ISO/IEC TR 15504-2, (1998). Information Technology – Software Process Assessment – Part 2 A reference model for process and processes capability, ISO and IEC, Geneva.

[Iso00] ISO 9001:2000, *Quality Management Systems-Requirements*: ISO, Third edition December 15, International Organization for Standardization, Geneva, Switzerland.

[Iso04] ISO/IEC 90003:2004, *Software and Systems Engineering-Guidelines for the Application of ISO9001:2000 to Computer Software*: ISO and IEC, 2004.

[Iti01a] Central Computer and telecommunications Agency. (2001) Service Support. In Information Technology Infrastructure Library. HSMO Books: London, UK;

[Iti01b] Central Computer and telecommunications Agency. (2001) Service Delivery. In Information Technology Infrastructure Library. HSMO Books: London, UK;

[Jef00] R.Jeffries, A.Anderson, C.Hendrikson, (2000) Extreme Programming Installed, Addison Wesley, December.

[Jon91] Jones, C. (1991) Applied Software Measurement. New York, NY:McGraw-Hill.

[Kaj01a] M. Kajko-Mattsson, (2001), "Corrective Maintenance Maturity Model", partial fulfillment of the requirements for P.H.D, report 01-015, Stockholm University (Sweden).

[Kaj01b] M. Kajko-Mattsson, S. Forssander, U. Olsson, (2001) "Corrective Maintenance Maturity Model: Maintainer's Education and Training", ICSE -International Conference on Software Engineering, IEEE Computer Society Press: Los Alamitos, CA.

[Kaj01c] Kajko-Mattsson, M., Forssander, S., Westblom, U. (2001). Corrective-Maintenance Maturity Model (Cm3): Maintainer's Education and Training ICSE, 610-619.

[Kaj01d] Kajko-Mattsson, M. (2001). Corrective Maintenance Maturity Model, partial fulfillment of the requirements for P.H.D, report 01-015, Stockholm University.

[Kaj01e] Kajko-Mattsson, M., Forssander, S., Olsson, U., (2001) Corrective Maintenance Maturity Model: Maintainer's Education and Training, in Proceedings, International Conference on Software Engineering, IEEE Computer Society Press: Los Alamitos, CA.

[Kaj01f] Kajko-Mattsson, M., Forssander, S., Andersson, G., Olsson, U., Developing CM^3: Maintainers' Education and Training at ABB, Journal of Computer Science Education, Swets&Zeitlinger, Vol. 12, No 1-2, pp. 57-89, 2002. (2001).

[Ker02] Kerzner, H. (2002) "Strategic Planning for Project Management Using a Project Management Maturity Model", John Wiley & Sons.

[Kra94] Krause, M.H. (1994) "Software - A Maturity Model for Automated Software Testing", Medical Devices & Diagnostic Industry Magazine, December issue.

[Leh80] L.L. Lehman, (1980) "Program Life-Cycles and Laws of Software Evolution", *Proceedings of IEEE*, vol. 68, n° 9, septembre 1980, p. 1060-1076.

[Leh85] Lehman, M.M. Belady, L.A. (1985) Program Evolution – Processes of Software Change, Publisher: London; Orlando: Academic Press Inc. Ltd. ISBN: 0124424406 0124424414.

[Leh97] Lehman, M.M. (1997) Laws of Software Evolution Revisited, EWSTP96, October, LNCS 1149, Spinger Verlag, pp 108-124.

[Lie78] B.Lientz, E.B. Swanson and G.E. Tompkins, (1978) *Characteristics of Applications Software Maintenance Comm. ACM,* Vol. 21.

[Lie80] Lientz, B. Swanson, E. (1980) Software Maintenance Management, Reading, Mass., Addison-Westley, 214 pp.

[Lud00] Ludescher, G., Usrey, M.W. (2000) "e-commerce Maturity Model", 1st international Research Conference on Organizational Excellence in the Third Millennium, R.Edgeman editor, Estes Park, CO, August 2000, 168-173.

[Luf01] Luftman, J. (2001), "Assessing Business-IT Alignment Maturity", Communications of AIS, 4(2).

[Mag97] S.Magee, L. Tripp, (1997) Guide to Software Engineering Standards and Specifications, Artech House, Boston-London.

[Mai02] Maintenance Technology, (2002) Using Benchmarking Data Effectively, [On Line] http://www.mt-online.com/current/03-01mm01.html (link tested on July 2nd 2004).

[Mal04] Malcolm Baldrige National Quality Program. (2004) Criteria for performance excellence, NIST, 70 pp. http://www.quality.nist.gov/PDF_files/2004_Business_Criteria.pdf (link tested on May 3rd 2004).

[Mar83] Martin, J. McClure, C. (1983) Software Maintenance: The Problem and its Solutions. Englewood Cliffs, NJ: Prentice-Hall, 472 pp.

[McC02] B.McCracken, (2002), "Taking Control of IT Performance", InfoServer LLC, Dallas, Texas, October.

[McG95] J.McGarry, (1995) Practical Software Measurement : A Guide to Objective Program Insight, DoD, Sept.

[Moo98] J.Moore (1998) Software Engineering Standards: A User's Road Map. IEEE CS Press.

[Mor96] S.Moriguchi, (1996) Software Excellence: A Total Quality Management Guide, Productivity Press.

[Mul02] Mullins, C. (2002) "*The Capability Model – from a data perspective*", The Data Administration Newsletter, [On line]. www.tdan.com/i003fe04.htm, (Link tested on July 2nd 2004).

[Nie02] F. Niessink, V. Clerk, H van Vliet, (2002) "*The IT Service Capability maturity Model*", release L2+3-0.3 draft, [On line]. http://www.itservicecmm.org/doc/itscmm-l23-0.3.pdf (Link tested on November 15th 2003).

[Osb90] W.M. Osborne, E.J.Chikofsky, (1990) Fitting Pieces to the Maintenance Puzzle, IEEE Software January, pp. 10-11.

[Pfl01] S.L. Pfleeger. (2001) *Software Engineering—Theory and Practice.* Prentice Hall, 2nd ed..

[Pia98] M.Piattini, F.Ruiz, M.Polo, J.Villalba, T. Batabchury, M.Martinez. (1998) Mantenimiento de software : conceptos, metodos, herramientas y outsourcing. RAMA. Madrid, Spain.

[Pig97] Pigoski T.M. *Practical software maintenance: Best practice for managing your software investment.* John Wiley & Sons: New York, NY, 1997; 384 pp.

[Pom02] Projxsoft, (2002) "*Project Organization Maturity Model* (POM2)", [On line]. http://www.projxsoft.com/default.asp?nc=2053&id=4 (Link tested on July 2nd 2004).

[Poo01] Poole, C., Huisman, W (2001). Using Extreme Programming in a Maintenance Environment, IEEE Software November/December, pp.42-50.

[Pre97] R.S. Pressman. Software Engineering: A practitioner's Approach. Fourth Edition, McGraw-Hill, fourth edition, 1997.

[Raf02] Raffoul, W. (2002), "The Outsourcing Maturity Model", Meta Group, [On line]. http://techupdate.zdnet.com/techupdate/stories/main/0%2C14179%2C2851971-2%2C00.html#level1 (Link tested on July 2nd 2004).

[Ray01] Rayner, P., Reiss, G. (2001), "*The Programme Management Maturity Model*", The Programme Management Group, 15th February, [On line]. http://www.e-programme.com/pmmm.htm (Link tested on July 2nd 2004).

[Sch01] Scheuing, A.Q., Fruhauf, K. (2000), "Maturity Model for IT Operations (MITO)", Swisscom AG, CH-3050 Bern. Switzerland, tel.: +41-31-342-8259, fax: +41-31-342-6086, email: Arnold.Scheuing@Swisscom.com

[She01] Sheard, S. (2001). The Frameworks Quagmire, Software Productivity Consortium [En ligne]. http://www.software.org/quagmire/ , (Kink tested on July 2nd 2004)

[Sch02] Schlichter, J. (2002), "*An Introduction to the emerging PMI Organizational Project Management Maturity Model"*, [On line]. http://www.pmi.org/prod/groups/public/documents/info/pp_opm3.asp (Link tested on July 2nd 2004).

[Sch00] Schmidt, M. (2000) Implementing the IEEE Software Engineering standards, Sams Publishing, Indianapolis, Indiana, October. ISBN: 0-672-31857-1, 242 pp.

[Sch87] Schneidewind, N.F. (1987) *The State of the Maintenance*. IEEE Transactions on Software Engineering; 13(3): 303-310.

[Sch99] Schmietendorf, A., Scholz, A. (1999), "The Performance Engineering Maturity Model at a glance", Metrics News, 4(2), Otto Von Guericke University of Magdeburg (Germany), ISSN 1431-8008, December issue, Magdeburg (Germany).

[Sei91] Paulk, M.C., Curtis, B., et al. (1991). Capability Maturity Model for Software, SEI, CMU/SEI-91-TR-24, August 1991.

[Sei93] CMM Product Development Team. (1993) Capability Maturity Model for Software (CMM), Version 1.1, CMU/SEI-93-TR-24, ESC-TR-93-177. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA; 64 pp.

[Sei02] CMMI Product Development Team. (2002) *Capability Maturity Model Integration for Software Engineering (CMMi)*, Version 1.1, CMU/SEI-2002-TR-028, ESC-TR-2002-028. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA; 707 pp.

[Sco88] Scott, T. Farley, T. (1988) Slashing Software Maintenance Costs, Business Software Review, indianapolis, March.

[Som97] Sommerville, I. Sawyer, P. (1997). "Requirements Engineering : A Good Practice Guide", John Wiley & Sons.

[Sri01] Sribar, V., Vogel, D. (2001), "The Capability Maturity Model for Operations", Metagroup, [On line] http://www.metagroup.de/metaview/mv0452/mv0452.html (Link tested on July 2nd 2004).

[Sta94] G.E.Stark, L.C.Kern, and C.V.Vowell. (1994) A Software Metric Set for Programme Maintenance Management. Journal of Systems and Software.

[Stp93] D.St-Pierre, (1993) Integration of Maintenance Metrics. in IEEE International Conference on Software Maintenance - ICSM-93. Montréal, Québec: IEEE.

[Str00] Stratton, R.W. (2000), 2The Earned Value Management Maturity Model2, [On line]. http://www.mgmt-technologies.com/evmtech.html (Link tested on July 2nd 2004).

[Swa89] Swanson, E.B., and Beath, C.M. (1989)., "Maintaining Information Systems in Organizations", John Wiley & Sons.

[Top98] Topaloglu, N.Y. (1998), "Assessment of Reuse Maturity", 5th International Conference on Software Reuse, Victoria (Casnada) June 2-5.

[Tri92] Trillium, (1992), "Model for the Telecom Product Development & Support Process Capability", Bell Canada, version 2.2, Montréal (Canada).

[Van00] Van Bon, J. (2000) World Class IT Service Management, Van Bon (eds.) ten Hagen & Stam Publishers : The Hague.

[Vee02] Veenendaal, V., Swinkels, R. (2002), "Guideline for testing maturity: Part 1: The TMM model", Professional Tester, Vol. three, Issue 1, [On line]. http://www.improveqs.nl/tmmart.htm (Link tested on July 2nd 2004)

[Vet99] Vetter, R. (1999), "The network maturity model for Internet Development", IEEE Computer, 132(10), 117-118.

[Wal94] D.S.Walton (1994), “Maintainability Metrics”, Centre for Software Reliability Conference, Dublin, City University, London UK.

[Wit99] Wichita State University, (1999), "Enterprise Engineering Presentation, Capability Model of BPR", course IE80I, Whichita.

[Zit95] Zitouni, M., Abran, A., Bourque, P. (1995) Élaboration d'un outil d'évaluation et d'amélioration du processus de la maintenance des logiciels: une piste de recherche, Le génie logiciel et ses applications, huitièmes journées internationales (GL95) , EC2 & Cie , Paris, La Défense , pp. 727-739.

Annex A: First Inventory and classification of Maintenance Processes and activities

Software Maintenance Process	Software Maintenance Activity	Software Maintenance Practice	ISO14764 [Iso98]	IEEE 1219 [Ieee98]	ISO12207 [Iso95]	SW CMMi [Sei01]	SWEBOK [Abr01]
Implementation Process	Process Definition	Identifier, Définir et Adapter les processus génériques du développement et de la maintenance pour l'organisation	s6.6, s7.2.2	A.3.1, A.3.5.1, A.3.5.2	s5.5.1, s5.3.1,	[Pa152.IG101.SP101]	s3.2.1, s3.2.1
		Etablir quelle organisation effectue la maintenance (développeur, maintenance, sous contrats, impartition)	s7.2.2			[GP106] seulement dans le cadre de la mesure [Pa166.IG101.SP103.SubP103.N102] seulement dans le cadre de sous-traitance	s3.3.2.4 s3.2.2.2.3
		Etablir un Programme de mesure de la maintenance (identifier, choisir, appliquer, valider et améliorer les mesures)	s6.5	table 2, A.12		[PA154.N101] [PA154.IG101] [PA154.IG101.SP102.N103] [PA154.IG102], [Pa154.IG101.SubP101 & SP102 & SP103 & SP104]	s3.3.4
		Etablir les définitions de taches, connaissances requises et plans de formation des employés de support, de la maintenance et des opérations		A.3.1		[GP106.SubP102] seulement dans le cadre du processus [GP107] [PA167.IG101.SP103.SubP106] dans le cadre du projet de développement	s3.1.1, s3.2.1, s3.2.2.2
		Etablir un processus d'amélioration des processus				[Pa152.IG101 & IG102] [GP117]	s3.3.4

		de la maintenance. Planifier et améliorer les processus					
	Document the maintenance activities	Etablir la procédure pour la réception de requêtes/rapports de problèmes, l'enregistrement et le suivi de requêtes de modifications (RM)/rapports de problèmes (RP)	s 8.1.2.2		s5.5.1.2	Rien de spécifique pour la maintenance	s3.2.1
		Etablir la procédure pour les activités de la Gestion de la Configuration du logiciel lors de la modification du logiciel	s 8.1.2.3	A.11	s5.5.1.3	[Pa159.N101] [Pa152.IG101.SP101] pour le projet de développement	s3.2.1
	Conformity control of process	Effectuer des revues qui assurent le suivi des processus de la maintenance	s 8.1.3	A.3.5.3		[GP113] [Pa145.IG101] [FM122.HDA103.T107]	s3.2.1
	Review software development activities that influence its maintainability	Dans le cas de fournisseurs externes, réviser et maintenir les clauses contractuelles qui touchent la maintenance			s5.5 (5.1.3)	[Pa166.IG101.SP103] [Pa166.IG102.SubP101.SubP106.N102] [Pa163.IG103.SP102.SubP104]	s3.2.1
		Revue du Plan de la Maintenance (plan crée par le développeur et revue par le groupe de la maintenance)	s 6.7, s6.8.1	A.7	s5.5 (5.1.4)	[Pa146.IG101.SP105.SubP101] [Pa166.IG102.SubP101.SubP107] [FM103.HDA101.HDB105.T104]	s3.2.1, s3.2.2.2.3
		Revue des spécifications de 'maintenabilité' établies par le développeur (avant la conception)	s6.8.2.1		s5.5 (5.1.4)	[Pa166.IG102.SP102] [FM103.HDA101.HDB105.T104]	s3.2.1
		Revue de la conception architecturale (modularité, analyse structurée,	s6.8.2.2		s5.5 (5.1.4)	[Pa166.IG102.SP102] [FM103.HDA101.HDB105.T104]	s3.2.1

		documentation adéquate)					
		Revue de la Conception détaillée	s6.8.2.3		s5.5 (5.1.4)	[Pa166.IG102.SP102] [FM103.HDA101.HDB105 .T104]	s3.2.1
		Revue de la programmation et essais unitaires	s6.8.2.4		s5.5 (5.1.4)	[Pa166.IG102.SP102] [FM103.HDA101.HDB105 .T104]	s3.2.1
		Revue des essais de qualification	s6.8.2.5		s5.5 (5.1.4)	[Pa166.IG102.SP103] [FM103.HDA101.HDB105 .T104]	s3.2.1
	Transition (from development to maintenance) Management	Etablissement du Plan de la maintenance incluant transition (Qui, rôles et resp. Comment, ressources, location et date de début, transfert des connaissances, formation, documentationl)	s6.7, s6.9, s7.1	A.3.5, Annexe C		[Pa166.IG102.SP104.WI01 &WI02, &WI03]	s3.2.2.2.3
		Revue de la documentation durant pour la maintenance (si nécessaire réviser/créer cette documentation)	s6.10		s5.5 (5.1.4)	[Pa166.IG102.SP102.SubP 104] [FM103.HDA101.HDB105 .T104] [Pa160.IG103.SP102.SubP 104 & SubP105]	s3.2.1
		Analyse des exigences de ressources en maintenance	s7.1, s7.4.2, s 7.4.3	A.3.5.5		[Pa166.IG102.SP104, SubP101]	s3.3.2.2
		Acceptation du logiciel par le groupe de la maintenance			s5.3.13 s5.5 (5.1.5)	[Pa166.IG102.SP103]	
		Effectuer la transition des produits de l'équipe de développement a l'équipe de maintenance	s6.9			[Pa166.IG102.SP104]	s3.3.2.4.2 contexte de outsourci ng
		Support de l'équipe de la maintenance par le			s5.3.13.3	[Pa166.IG101.SP103.SubP 103]	

		développeur pour une période déterminée				seulement dans le cadre de contrats avec un fournisseur	
Analysis of Problem Modifications	Identify and route modification requets (MRs) and problem reports (PRs)	Assigner uniquement chaque requête (avec un identifiant) avec l'outil de gestion du bureau d'aide et dépêche la requête au groupe pertinent.		s4.1.2		[PA159.IG101.SP102] [PA159.IG101.SP102.N102] [PA159.IG102] seulement si on traite les requêtes comme des demandes de changements pour un projet	s3.2.2.2.1
	Classify request according to a maintenance category	Classer par le type de travail de maintenance requis (Corrective, Adaptive, Perfective ou Préventive)	s4.3, s4.9, s4.10, s4.11	s4.1.2, s3.1.1, s3.1.2, s3.1.7, A.4.1			s3.1.6
	Investiguer the validity of the request	Vérifie/Reproduire les RM/RP afin établir leur légitimité (accepte/rejette)	s 8.2.2.1	s4.1.2	s5.5.2.2		s3.2.1
	Priorise requests	Etablir la priorité relative de chaque requête avec la clientèle		s4.1.2			
	Estimate effort of requests	Effectuer un estimé de l'effort requis pour le travail à faire	s7.2.4	s4.1.2		[Pa163.IG101.SP102&SP104]	s3.3.3
	Assign request to individuals	Etablir quel membre de l'équipe va travailler sur quelle requête 'staffing study'	s7.2.3, s7.4.1	s4.1.2		[Pa163.IG102.SP104.SubP102.N102] seulement avec l'utilisation de techniques de gestion de projet	s3.3.4.1
	Analysis of MRs and PRs	Effectuer une étude 'haut niveau' des exigences et d'impact du changement ou de la correction d'un problème	s 8.2.2.1	s4.2.2, A.4.2	s 5.5.2.1	Niv. 2 [Pa146.IG101] Niv. 3 [Pa157.IG101 to IG103] [PA159.IG102.SP101.SubP102] seulement si on traite les requêtes comme des demandes de changements	s3.2.1

						des exigences pour un projet	
		Etablir des alternatives de solutions 'haut niveau'	s 8.2.2.3	s4.2.2.1	s 5.5.2.3	[Pa160.IG101] seulement si on traite les requêtes comme des exigences dans un projet	s3.2.1
		Documenter le RM/RP, le résultat de l'analyse et les options de solutions	s 8.2.2.3		s 5.5.2.4	[PA160.IG101.SP104.SubP 106] [PA159.IG102.SP101.SubP 101] seulement si on traite les requêtes comme des demandes de changements pour un projet	s3.2.1
		Evaluer l'impact sur la conversion de données, sur la sûreté et sécurité, sur les facteurs humains et sur les coûts/ bénéfice (court terme et long terme) de la proposition		s4.2.2.1, A.9, A.10		[Pa163.IG102.SP102.W103 & W105] [PA159.IG102.SP101.SubP 102] seulement si on traite les requêtes comme des demandes de changements pour un projet	s3.4.4
		Identifier, analyser, prioriser et adresser les risques associes à la proposition	s 8.2.2.5	s4.4.3, s4.4.2.3, A.8		[Pa163.IG102.SP103] seulement si on traite la proposition comme une proposition de projet	s3.2.2.1 s3.3.1.3
		Obtenir l'approbation avant d'effectuer l'implantation de la proposition de modification (incluant le client)	s 8.2.2.5		s 5.5.2.5	[Pa163.IG103.SP102.SubP 102.N101] [Pa163.IG102.SP103.SubP 103]	s3.2.1
Implantation de modification	Detailed Analysis of a request	Transformer les exigences en spécifications techniques détaillées		s4.2.2.2	s5.5.3.2 (s 5.3.2, s5.3.3,	[Pa157.IG103]	s3.2.1

	(MRs/PRs)				s5.3.4)		
		Identifier les composants a modifier (quels documents, programmes, structures de données,…)	s 8.3.2.1	s4.2.2.2	s5.5.3.2 (s 5.3.4.1, s 5.5.2.5)	[Pa157.IG103.SP102]	s3.4.3 s3.4.4
		Identifier les interfaces impactées	s 8.3.2.1			[Pa157.IG103.SP103]	s3.1.5
		Spécifier les aspects de sûreté et sécurité impactes		s4.2.2.2	s5.5.3.2 (s 5.3.4.1)	[Pa157.N101] [Pa163.IG101.SP102.SubP 101.N101] [Pa163.IG102.SP102]	s3.1.6
		Etablir la stratégie d'essais		s4.2.2.2	s5.5.3.2 (s 5.3.5.5)		s3.3.1.2
	Detailed Analysis control and follow-up	Gestion de la Configuration : (Création d'une configuration de référence des exigences et de la solution approuvées par le client)		s4.2.3, A.11.2	s5.5.3.2 (s 5.3.4.3)	[Pa159.N110] [Pa159.IG101, & IG102]	s3.2.2.2.1
		Revue technique des spécifications détaillées, plans de test et estimes		s4.2.3		[FM114.HDA103.HDB104 .T101]	s3.2.1 s3.2.2.2
	Design associated with a request (MRs/PRs)	Transformer les spécifications en conception et identifier les composantes impactées		s4.3.2, A.4.3	s5.5.3.2 (s5.3.5, s5.3.6)	[Pa160.IG102]	s3.2.2.1 s3.3.1.3
		Initier les modifications de la documentation technique et client des composantes impactées		s4.3.2	s5.5.3.2 (s5.3.6.4)	[Pa160.IG101.SP104.N102] [Pa160.IG102.N101]	s3.2.2.1 s3.3.1.3
		Créer un plan de test et jeu d'essais		s4.3.2	s5.5.3.2 (s5.3.6.5)	[Pa150.IG101.SP101.N102] [Pa150.FL104 & EC111]	
	Design control and follow-up	Inspection de la conception détaillée		s4.3.3	s5.5.3.2 (s5.3.6.8)	[Pa147.IG102.SP101] [FM102.HDA104.HDB102	s3.2.1 s3.2.2.2

						.T115]	
		Gestion de la Configuration (Réservation du code source et de la documentation qui fera objet de modification)		s4.3.3, A.11.3		[Pa159.IG102.SP101.SubP 103]	s3.2.2.2.1
		Vérifier la documentation et Vérifier la 'tracabilité' des spécifications <->conception		s4.3.3	s5.5.3.2 (s5.3.6.7) (s5.3.6.8)	[Pa146.IG101.SP104]	s3.2.2.2.2
	Implementation the change in the software and documentation	Programmation & Essais Unitaires		s4.4.2.1, A.4.4	s5.5.3.2 (s 5.3.7)	[Pa160.IG103] [Pa160.IG103.SP101.SubP 104]	s3.2.1
	Implementation control and follow-up	Inspection des programmes et revue des tests/documentation		s4.4.3, s4.4.2.4	s5.5.3.2 (s 5.3.7.3)	[Pa160.IG103.SP101.SubP 103 &SubP105 & SubP102.AMP102] [FM102.HDA104.HDB102 .T117]	s3.2.2.2.2
		Vérifier la gestion de la configuration		s4.4.3, A.11.4	s5.5.3.2 (s 5.3.10.1)	[Pa163.IG103] [GP109], [PA160.EL106], [PA159.IG102.SP102.SubP 104]	s3.2.2.2.1
		Vérifier la 'tracabilité' conception détaillée ←>programmes		s4.4.3	s5.5.3.2 (s 5.3.7.5)	[PA160.N104] [PA160.IG102.SP103.SubP 101.N101]	
	Product Integration	Préparation de l'environnent d'essais d'intégration et conduite des essais d'intégration des unités modifiées	s 8.3.2.2	s4.4.2.2 s4.5.2, A.4.5	s5.5.3.2 (s 5.3.8) (s 5.3.10)	[Pa147.IG103.SP103]	
		Préparation de l'environnent d'essais Fonctionnels et conduite des essais fonctionnels	s 8.3.2.2	s4.5.2	s5.5.3.2 (s 5.3.10)	[Pa147.IG103.SP103] [Pa150.IG101 & 103] [Pa150.IG103.SP102]	
		Essais de Régression	s 8.3.2.2	s4.5.2	s5.5.3.2 (s 5.3.10)	[Pa147.IG103.SP103] [Pa159.IG102.SP102.SubP	s3.3.1.2

						104]	
	Integration control and follow-up	Revue des essais de système et évaluer notre préparation pour activité d'acceptation utilisateur		s4.5.2	s5.5.3.2 (s 5.3.11.4)	[GP110, Pa147.EL112] [Pa150.IG102] [Pa149.IG103.SP102] [Pa147.IG101.SP101.SubP 106] [PA149.W106] [PA149.IG101]	s3.2.1 s3.2.2.2
Review/Accept the change	Acceptance Test of change	(AFU) Essais d'Acceptation Fonctionnelle du représentant des Utilisateurs	s 8.4.2.1	s4.6.2, A.4.6	s5.5.3.2 (s 5.3.11) (s5.3.13.1), s5.5.4.2	[Pa149.IG102.SP101] [Pa149.N102]	s3.2.1
		Rapporter les résultats et corriger les problèmes	s 8.4.2.1	s4.6.3	s5.5.3.2 (s5.3.13.1), s 5.5.4.2	[Pa149.IG102.SP102] [PA149.EL115]	s3.1.1 s3.2.1
		Etablir la configuration finale après acceptation (gestion de la configuration)	s 8.4.2.1	s4.6.3, A.11.5, A.11.6	s5.5.3.2 (s 5.3.13.1)	[Pa159.IG101] [Pa149.EL105] [Pa159.IG103.SP102.SubP 103 & 104]	s3.2.2.2.1
	User sign-off	Obtenir l'approbation finale du représentant des utilisateurs	s 8.4.2.2		s5.5.3.2 (s5.3.13.1) s5.5.4.2	[GP124.PA150.EL113 & GP112]	s3.2.1
	Move to production	Développer un plan d'implantation de la solution		s4.2.2.2, A.3.5.6	s5.5.3.2 (s 5.3.12.1)	[Pa147.IG101.SP101.SubP 106.N101] [PA147.EL102]	s3.2.1
	Delivery	Réviser la configuration finale		s4.7.2, A.4.7		[GP109] [PA147.IG103.SP101.SubP 105]	s3.2.2.2.1
		Informer les utilisateurs de la date d'entrée en vigueur du changement		s4.7.2			s3.2.1
		Archiver l'ancienne version		s4.7.2,		[PA159.IG101.SP102.SubP	

		(créer une copie de recouvrement)		A.11.7		107.N101]	
		Installer la nouvelle version, (opérer en parallèle) et informer les utilisateurs		s4.7.2	s5.5.3.2 (s5.3.12.2)	[PA147.IG103.SP104.SubP 106]	s3.2.1
		Former ou répondre aux questions concernant les changements		s4.7.2			
	Delivery control and follow-up	Documenter la configuration finale		s4.7.3		[Pa147.IG103.SP104.SubP 101]	s3.2.2.2.1
		Procurer le matériel (documentation, information…) aux utilisateurs		s4.7.3		[Pa147.IG103.SP104.SubP 102] [Pa147.IG103.SP104.w102] [Pa160.IG103.SP102.W101]	
		Mettre a jour la bases de données du groupe de maintenance concernant cette requête (efforts, fermeture de la requête)		s4.7.3		[Pa159.IG101.SP103] [PA159.IG102.SP101.SubP 104.N101] seulement si on traite les requêtes comme des demandes de changements pour un projet	
		Effectuer les étapes de fermeture de gestion de la configuration et documentation		s4.7.3		[Pa159.IG103]	s3.2.2.2.1
Migration		Migration d'un logiciel d'une plateforme a une autre	s 8.5		s5.5.5		s3.2.1 s3.2.2.2.1 , s3.1.1
Retraite		Retraite d'un logiciel	s 8.6	A.13	s5.5.6		s3.1.1, s3.2.1

Annex B: Enhanced classification of Software Maintenance Processes and activities

Software Maintenance Process Domains	Software Maintenance Key Process Areas	Software Maintenance Practices	Zitouni [Zit95]	Corrective-Maintenance Maturity Model [Kaj01d]	Camélia [Cam94]	IT Service CMM [Nie02]	Cobit [Cob00]
Mainenance Process Management	**Maintenance Organizational Process Focus**	Etablir un processus d'amélioration des processus de la maintenance. Planifier et améliorer les processus	MCO.12/01 a MCA4.12/01 (10 pratiques niveau2 a 4)		3.3.2.1 a 3.3.5.1 (25 pratiques niveau 2 a 5)	6.3 (3 buts, 3 obligations, 4 habilité, activités 1, 2, 3, 4 et 7)	M1, M2, PO8
	Maintenance Process definition	Identifier, Définir et Adapter les processus génériques du développement et de la maintenance pour l'organisation	MCO.13/01 a MCA3.13/06 (8 pratiques niveau2 et 3), MCA3.16/02		3.1.2.1 a 3.1.4.1 (14 pratiques niveau 2 a 4)	6.1 (2 buts, 1 obligation, 3 habilité, 4 activités)	AI2, AI4, PO8
		Documenter les Activités : Etablir la procédure pour la réception de requêtes/rapports de problèmes, l'enregistrement et le suivi de requêtes de modifications (RM)/rapports de problèmes (RP)			4.1.3.19 dans le cadre du projet de développement	6.2 (haut niveau)	AI4
		Etablir la procédure pour les activités de la Gestion de la Configuration du logiciel lors de la modification du logiciel			6.7.2.1, 6.7.2.2,	5.5 (obligation 1)	DS9
		Etablir quelle organisation effectue la maintenance (développeur, le groupe de maintenance, sous contrats, impartition)			1.2.2.5	5.2 (indirect via la planification des services)	PO4, PO7
	Maintenance Training	Qualification et formation du personnel	MCO.14/01 a SBA4.14/03 (8 pratiques niveau 2 a 4)		2.1.2.1 a 2.1.3.17 (18 pratiques	6.3 Activité 6, 6.7	PO7

					niveau 2 a 3)		
		Etablir les définitions de taches, connaissances requises et plans de formation des employés de support, de la maintenance et des opérations			1.2.2.4, 1.2.3.1, 2.1.3.1 a 2.1.3.17	5.2 (indirect via la planification des services)	PO7
	Organizational process performance	Etablir un Programme de mesure de la maintenance (identifier, choisir, appliquer, valider, analyser et améliorer les mesures) et établir le SLA avec le client			3.4.3.1, 4.5.2.1 a 4.5.3.4 (8 pratiques niveau 2 et 3)	Mesures individuelles dans tous les key process areas du modèle, SLA dans la section 5.1	M1, M2, DS1, PO8
	Innovations and their deployment	Les nouvelles technologies sont évaluées et introduites d'une manière contrôlée	RBA3.16/08 a MCA4.16/01 (10 pratiques de niveau 3 et 4) MCO.20/01 a MCA3.20/04 (10 pratiques niveau 2 et 3)		*3.2.2.1 a 3.2.4.2 (8 pratiques niveau 2 a 4)	6.3 activité 5, 6.5 activité 1	AI3, AI1, AI6, DS12
		Environnement de développement et d'entretien	GEA3.16/01		7.1.2.1 a 7.1.5.1	6.5 activité 1	AI1, AI6
Maintenance Request Management	**Maintenance Planning**		GEO.02/01 a GEA5.02/03 (43 pratiques de niveau 2 a 5)		[4.1.2.5,4.1.2.6,4.1.2.11, 4.1.2.12, 4.1.2.22, 4.1.3.7 dans le cadre du projet de développement] 8.5.2.1	5.2 (3 buts, 2 obligations, 3 habilité, 10 activités)	PO5
	Control and follow-up of		MCO.03/01 a MCA4.03/02		4.1.2.8,4.1.2.13 a	5.3 (3 buts, 2 obligations, 14	PO6

	maintenance work		(25 pratiques de niveau 2 a 4)		4.1.2.17, 4.1.2.19 a 4.1.2.21a 4.1.2.23 dans le cadre du projet de développement	activités)	
	Su**pplier and contract Management**	Contrat, travail conjoint et contrôle des livrables du fournisseur	GEO.04/01 a MCA3.04/03 (19 pratiques de niveau 2 a 4)		4.2.2.1 a 4.2.3.1, 4.3.2.1 a 4.3.3.6 (24 pratiques)	5.4 (4 buts, 2 obligations, 3 habiletés, 12 activités)	DS2
	Problem Management		GEO.08/01 a RBA4.08/01 (13 pratiques de niveau 2 a 4)		8.1.2.1 a 8.1.2.3	6.9 (3 buts, 1 obligation, 3 habileté, 9 activités)	DS10
		Identification et dépêche des requêtes de modification (RM) et rapport de problème (RP): Assigner uniquement chaque requête, sa source (avec un identifiant) avec l'outil de gestion du bureau d'aide et dépêche la requête au groupe pertinent.		TA-Resource-1, 2 & 4. TA-Process-PAC-6 a 14		6.9 activité 5.2	
		Classification de la requête selon les catégories de la maintenance : Classer par le type de travail de maintenance requis (Corrective, Adaptive, Perfective ou Préventive)		TA-Process-PAC-1 et 2	4.1.3.19	6.9 activité 5.3	
		Investiguer la légitimité d'une requête : Vérifie/Reproduire les RM/RP afin établir leur		TA-Product-1, TA-Process-	8.1.2.2		

		légitimité (accepte/rejette)		PI-2, TA-Process-PDR-1 a 6			
		Prioriser les requêtes : Etablir la priorité relative de chaque requête avec la clientèle		TA-Process-PAC-3, 4, 5	4.1.2.18	6.9 activités 5.4 et 5.5	DS4
		Système de réponse aux problèmes			8.1.2.1 a 8.1.3.2 (6 pratiques de niveau 2 et 3)	6.9 activité 4	DS4
		Ingénierie orientée client			8.5.2.2		
	Quantitative Management		MCO.17/01 a SBA4.17/12 (16 pratiques niveau 2 a 4)		4.1.2.18	Mesures individuelles dans tous les key process areas du modèle, SLA dans la section 5.1	
		L'efficacité des essais de la vérification est mesurée et suivie conformément à une procédure documentée			6.6.4.1		
		L'efficacité des essais de la vérification est mesurée et suivie conformément à une procédure documentée			6.6.4.2		
		La densité des défauts est planifiée et comparée avec la densité réelle des défauts			6.6.4.3		M1
Evolution Engineering	**Requirement Management**	Analyse détaillée d'une requête (RM/RP): Transformer les exigences en spécifications techniques détaillées		TA-Product-8, TA-Process-PI-1 a 3	4.4.2.1 a 4.2.3.1 (11 pratiques niveau 2 a 3)	6.5 activité 2 (très haut niveau), 6.9 activité 5.7 (RP)	
		Assigner une requête a une ressource : Etablir quel	MCA3.01/02	TA-Resource-3	4.1.2.2 dans le cadre du	6.5 habileté 1 (très haut niveau)	

		membre de l'équipe va travailler sur quelle requête 'staffing study'			projet de développement		
		Analyse d'une requête (RM/RP) – Gestion des exigences : Effectuer une étude 'haut niveau' des exigences et d'impact du changement ou de la correction d'un problème		TA-Process-PCI-1 a 3, TA-Process RCA-1 a 3	4.1.2.5 [dans le cadre du projet de développement] 4.4.2.1 a 4.4.3.4 (14 pratiques) 7.1.2.5	6.5 activité 2 (très haut niveau)	
		Établir des alternatives de solutions 'haut niveau'		TA-Process-MD-1 a 3			
		Effectuer un estimé de l'effort requis pour le travail à faire		TA-Process-MD-4, TA-Process-MD-5.3, 5.5	4.5.2.1 a 4.5.3.4 (8 pratiques de niveau 2 et 3)		DS6
		Documenter le RM/RP, le résultat de l'analyse et les options de solutions	GEA2.01.01	TA-Process-MD-5.4	6.5.2.1, 6.5.3.1, 6.5.3.5, 6.5.3.6, 6.5.3.7,		
		Évaluer l'impact sur la conversion de données, sur la sûreté et sécurité, sur les facteurs humains et sur les coûts/ bénéfice (court terme et long terme) de la proposition		TA-Process-MD-7	6.4.2.1 a 6.4.4.2 (25 pratiques d niveau 2 et 3), 4.4.2.2		DS5, DS11
		Identifier, analyser, prioriser et adresser les risques associes à la proposition		TA-Process-MD-5.6, 6 et 7	4.1.2.10, 4.1.3.12 [dans le cadre du projet de développem		PO9

					ent]		
		Obtenir l'approbation avant d'effectuer l'implantation de la proposition de modification (incluant le client)	MCA2.01/02	TA-Process-DM-1 a 5	4.1.2.1 [dans le cadre du projet de développement]		
		Identifier les composants a modifier (quels documents, programmes, structures de données,…)		TA-Process-MD-3.1			DS11
		Identifier les interfaces impactées		TA-Process-MD-3.1			
		Spécifier les aspects de sûreté et sécurité impactes		TA-Process-MD-3.1	4.4.2.2		DS5
	Technical Solution	Conception détaillée d'une requête (RM/RP) : Transformer les spécifications en conception et identifier les composantes impactées		TA-Process-MD-5.2	6.5.2.2, 6.5.3.2,	6.5 activité 2 (très haut niveau)	
		Initier les modifications de la documentation technique et client des composantes impactées		TA-Process-MI-1	6.5.2.1 a 6.5.3.7 (11 pratiques niveau 2 et 3	6.5 activité 2 (très haut niveau)	
		Implantation d'un changement dans le logiciel et la documentation : Programmation & Essais Unitaires		TA-Process-MI-1	6.5.2.3, 6.5.3.3	6.5 activité 2 (très haut niveau)	
	Product Integration	Intégration du Produit : Préparation de l'environnent d'essais d'intégration et conduite des essais d'intégration des unités modifiées		TA-Process-MD-5.1			PO11
	Verification	Préparation de l'environnent d'essais Fonctionnels et		TA-Product-5	6.6.2.3		PO11

		conduite des essais fonctionnels					
		Etablir la stratégie d'essais	RBA2.07/04		6.6.2.1		PO11
		Revue technique des spécifications détaillées, plans de test et estimés		TA-Process-MD-5	4.1.2.31[dans le cadre du projet de développement], 6.6.2.2		PO11, M3
		Créer un plan de test et jeu d'essais			6.5.2.4, 6.5.3.4,		PO11
		Contrôle de la conception détaillée : Inspection de la conception détaillée		TA-Process-MD-6			PO11
		Vérifier la 'tracabilité' conception détaillée ←>programmes	MCA3.16/07	TA-Product-7	6.7.2.14		PO11
		Essais de Régression			6.6.2.5		PO11
		L'exécution des jeux d'essais est automatisée		TA-Process-MI-1.2.2	6.6.3.2		PO11
		Contrôle de l'intégration : Revue des essais de système et évaluer notre préparation pour activité d'acceptation utilisateur	MCA3.16/03,		4.1.2.31 [dans le cadre du projet de développement], 6.6.3.5		PO11, M3
		Contrôle de l'implantation : Inspection des programmes et revue des tests/documentation		TA-Process-MI-1.2.3	6.6.3.3,		PO11
	Validation	Essais d'acceptation : (AFU) Essais d'Acceptation Fonctionnelle du représentant des Utilisateurs	GEA2.16/02 MCA3.16/05, RBA2.07/01, RBA2.07/06, RBA2.07/10		6.6.2.4		AI5, M3
		Contrôle d'AFU:Rapporter les résultats et corriger les problèmes	MCA3.16/04		6.6.3.5	6.5 activité 6 (très haut niveau)	AI5

		Approbation client : Obtenir l'approbation finale du représentant des utilisateurs	RBO.07/03		8.5.2.7, 4.1.2.18		AI5
		Implantation : Développer un plan d'implantation de la solution		TA-Product-6	8.5.2.7	6.5 activité 3 (très haut niveau)	
		Informer les utilisateurs de la date d'entrée en vigueur du changement			8.5.2.7		DS8
		Installer la nouvelle version, (opérer en parallèle) et informer les utilisateurs	GEA2.16/02		8.5.2.8, 8.5.3.5	6.5 activité 4 et 5 (très haut niveau)	
		Former ou répondre aux questions concernant les changements			8.6.2.1 a 8.6.3.2 (4 pratiques de niveau 2 et 3)		DS7, DS8
		Procurer le matériel (documentation, information…) aux utilisateurs	MCA3.16/06		8.4.2.1 a 8.4.4.2 (10 pratiques de niveau 2 a 4), 8.5.2.3		DS7, DS8
		Mettre a jour la base de données du groupe de maintenance concernant cette requête (efforts, fermeture de la requête)			4.1.2.24	5.3 activité 6	DS6
	Transition Management	Gestion de transition d'un logiciel spécifique du développement vers la Maintenance : Etablissement du Plan de la maintenance incluant transition (Qui, rôles et resp. Comment, ressources, location et date de début, transfert des connaissances,	RBO.07/01, RBA2.07/02			6.6 (3 buts, 1 obligation, 5 habiletés, 6 activités)	

		formation, documentation du logiciel et du matériel)					
		Revue du Plan de la Maintenance (plan crée par le développeur et revue par le groupe de la maintenance)			4.1.2.26 [dans le cadre du projet de développement]	6.6 activité 4 et 6	
		Revue des spécifications de 'maintenabilité' établies par le développeur (avant la conception)	RBA2.07/05		6.9.2.1 a 6.9.4.1 (11 pratiques de niveau 2 a 4)	6.6 activité 4 et 6	
		Revue de la conception architecturale (modularité, analyse structurée, documentation adéquate)			6.6.2.2, 6.6.3.3	6.6 activité 4 et 6	
		Revue de la Conception détaillée			6.6.2.2, 6.6.3.3	6.6 activité 4 et 6	
		Revue de la programmation et essais unitaires			6.6.2.2, 6.6.3.3	6.6 activité 4 et 6	
		Revue des activités du développement qui influence la 'maintenabilité' d'un logiciel spécifique : Dans le cas de fournisseurs externes, réviser et maintenir les clauses contractuelles qui touchent la maintenance			6.6.2.2, 6.6.3.3	6.6 activité 4 et 6	
		Revue des essais de qualification			6.6.2.2, 6.6.3.3	6.6 activité 4 et 6	
		Revue de la documentation durant pour la maintenance (si nécessaire réviser/créer cette documentation)			4.1.2.29 [dans le cadre du projet de développement]	6.6 activité 4 et 6	

		Analyse des exigences de ressources en maintenance			4.4.2.2	6.6 activité 3	
		Acceptation du logiciel par le groupe de la maintenance	RBO.07/02, RBA2.07/07			6.6 activité 4	
		Effectuer la transition des produits de l'équipe de développement a l'équipe de maintenance	GEA2.16/01, RBA2.07/03			6.6 activité 4	
		Support de l'équipe de la maintenance par le développeur pour une période déterminée	RBA2.07/08, 09, RBA2.07/11				
Support	**Configuration Management**		MCO.06/01 a RBA3.06/07 (16 pratiques de niveau 2 a 3)		6.7.2.1 a 6.7.5.1 (22 pratiques de niveau 1 a 5)	5.5 (4 buts, 1 obligation, 4 habiletés, 10 activités)	
		Contrôle de l'analyse détaillée : Gestion de la Configuration : (Création d'une configuration de référence des exigences et solution approuvées par le client)	MCA2.06/02		6.7.2.9	5.5 activité 4 et 6	
		Gestion de la Configuration (Réservation du code source et de la documentation qui fera objet de modification)		TA-Product-5	6.7.2.13	5.5 activité 4	
		Vérifier la gestion de la configuration	MCA3.06/04		6.7.2.12	5.5 activité 10	
		Vérifier la documentation et Vérifier la 'tracabilité' des spécifications <->conception	MCA3.16/07	TA-Product-7	6.7.2.14, 6.7.3.2	5.5 activité 10	
		Etablir la configuration finale après acceptation (gestion de la configuration)			6.7.2.9	5.5 activité 7	
		Vérifier la documentation et Vérifier la 'tracabilité' des	MCA3.16/07	TA-Product-7	6.6.2.2		

		spécifications <->conception					
		Archiver l'ancienne version (créer une copie de recouvrement)			6.7.2.5, 6.7.2.9	5.5 activité 7	
		Livraison : Réviser la configuration finale			4.1.2.31 [dans le cadre du projet de développement]	5.5 activité 10	
		Effectuer les étapes de fermeture de gestion de la configuration et documentation	MCA3.06/04		6.7.2.5, 6.7.2.9	5.5 activité 7	
	Quality Assurance	Qualité des processus organisationnels			1.1.2.1 a 1.1.4.5 (19 pratiques niveau 2 a 4)		PO11
		Système Qualité et assurance qualité	MCO18/01 a RBA4.18/07 (19 pratiques niveau 2 a 4), MCO.05/01 a SBA3.05/01 (19 pratiques de niveau 2 a 4)		5.1.2.1 a 5.1.5.1 (25 pratiques niveau 2 a 5)	5.7 Assurance qualité indépendante seulement	PO11, M3
		Effectuer des revues qui assurent le suivi des processus de la maintenance			5.1.2.5	6.2 vérification 1	M4
		Ingénierie des processus d'affaires			1.2.2.1 a 1.2.5.1 (24 pratiques niveau 2 a 5)		
	Causal analysis		MCO19/01 a MCA5.19/02 (11 pratiques		3.3.3.10, 3.3.3.11, 5.1.3.4		

			de niveau 2 a 5)				
	Reengineering/ Reverse Engineering				6.10.2.1 a 6.10.4.5 (17 pratiques de niveau 2 a 4) 7.1.2.4, 7.1.2.7, 7.1.2.8, 7.1.2.9		
	Reuse				6.8.2.1 a 6.8.4.1 (9 pratiques de niveau 2 a 4)		