SM^{CMM} Model to Evaluate and Improve the Quality of Software Maintenance Process: Overview of the model

(Alain April[1], Alain Abran[1], Reiner R. Dumke[2])
[1]École de Technologie Supérieure, Montréal, Canada, aapril@ele.etsmtl.ca , aabran@ele.etsmtl.ca
[2]Otto von Guericke University of Magdeburg, Germany, dumke@ivs.cs.uni-magdeburg.de

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 revised version of a maintenance-specific evaluation model: Software Maintenance Capability Maturity Model (**SM**^{CMM}). 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**^{CMM} is based on practitioners' experience, international standards and the seminal literature on software maintenance.*

1. Introduction

While multiple solutions to problems of software development have been proposed (such as development methodologies, management models and software tools) the function of software maintenance has not attracted such attention despite its significant share of the software budget in organizations (between 50% and 70%). Maintenance still suffers from a scarcity of best practice proposals that could readily be applied in the industry. Aside from Kajko-Mattsson [*Kaj01a*] recent proposal of an evaluation model specific to corrective maintenance, there is still a large number of software maintenance best practices that need to be recognized and better described for technology transfer to the industry at large. The inadequate share of management attention that software maintenance receives, and the fact that it suffers from lack of planning, are illustrated by the crisis management style typically adopted in this domain, coupled with the fact that software maintenance is still perceived as being expensive and ineffective.

For the software development function, there already exist many management models to evaluate the quality of the development process and to propose improvements. However, for the software maintenance function, there is a lack of published comprehensive models, which takes into account the characteristics specific to the maintenance process.

Recognizing the importance of software maintenance and the limitations of the process assessment models that emphasize development over maintenance, an initial draft of a comprehensive maintenance evaluation model was published in 1996 [*Zit96*]. This paper presents an update of the 1996 version of the Software Maintenance Capability Maturity Model – (**SM**^{CMM}) and documents and traces to it from other models.

Section 2 presents the findings and contributions from the literature, including a discussion of what is missing in the CMMi© to reflect a maintainer's point of view. Section 3 presents a comprehensive model of the software maintenance interfaces, followed by the base design of the proposed **SM**^{CMM}. Section 4 presents the high-level process model resulting from literature review. Section 5 describes the model and architecture of version 2 of **SM**^{CMM} as well as the generic criteria for each level of maturity. This is followed in section 6 by an example of the content of a specific process area (Management of Service Requests and Events).

[1] CMMi is a trademark of the SEI.

Finally, current work in progress is presented in section 7.

2. Prior Contributions

2.1 Researchers contributions

A literature search has not resulted with any comprehensive diagnostic techniques to evaluate the quality of the maintenance process of a given organization, nor to identify an improvement path. Table 1 presents an inventory of recent 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 inventory, only a handful (shown in **bold** in table 1) include 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 body of knowledge specific to software maintenance. This body of knowledge has recently been documented in the chapter 6 of the Guide to the Software Engineering Body of Knowledge [*Abr04*].

Year	Software Engineering CMM proposals
1991	Boo91
1992	**Tri92**
1993	Sei93
1994	**Cam94**, Kra94
1996	Bur96 & 96a, **Zit96**, Dov96
1997	Som97
1998	Esi98, Top98, Baj98
1999	Wit99, Vet99, Sch99
2000	**Cob00**, Str00, Bev00, Lud00
2001	**Kaj01a & 01b**, Ray01, Sch01, Luf01, Tob01, Sri01
2002	**Sei02**, **Nie02**, Mul02, Vee02, Pom02, Raf02, Sch02, Ker02, Cra02

Table 1: Software Engineering CMM proposals, sorted by year of publication

[*Apr04*] presents the mapping of a much larger number of software maintenance references: a) standards; b) relevant software engineering CMM proposals; and c) recognized software maintenance references. From these mappings, a large number of software maintenance best practices have been identified and listed. The key references presented are:

- The software maintenance standards ISO12207 [*Iso95*], ISO14764 [*Iso98*] and of IEEE1219 [Iee98];
- The most widely recognized quality models ISO9000-3: 2000 [*Iso00*] and the CMMi© [*Sei02*];
- Process evaluation model standard ISO/IEC TR 15504 (SPICE) [*Iso98a*];

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

- Cm3-Corrective Maintenance Model [*Kaj01a*];
- Cm3-Maintainer's Education Model [*Kaj01b*];
- ITIL Service Support [*Iti01a*];
- ITIL Service Delivery [*Iti01b*];
- IT Service CMM [*Nie02*];
- CobIT [*Cob00*];
- Malcolm-Baldrige [*Mal03*];
- *Camélia* Maturity Model [*Cam94*];
- ***SM***^{CMM} version 1 [*Zit96*].

Some of the ***SM***^{CMM} model improvements had been documented in [Apr01, *Apr02*], and experimented in a Middle East phone company. Another refinement is derived from the CMMi© [*Sei02*] adoption of the continuous representation, while the continuous representation itself can be traced back to its successful use in the past by other models such as: Bootstrap [*Boo91*] and *Camélia* [*Cam94*] just to name a few. These improvements to ***SM***^{CMM} have provided the following benefits: a) inclusion of Spice recommendations; b) a more granular rating for each roadmap and domain; and c) identification of specific practices across maturity levels, together with a path from level zero (absent) to a higher level of maturity. Furthermore, ***SM***^{CMM} has been aligned to the CMMi© model and to many of best practices documented in the software maintenance literature.

2.2 CMM© and CMMi© Models

The initial version of the model [*Zit96*] included only, in its literature review, two references (a) [*Swa89*]; and b) [*Ball90*]). Version 2 of the ***SM***^{CMM} has benefited from a much larger number of references, each carefully reviewed to ensure a wider and more representative

coverage of the maintenance processes. This review has also confirmed that some maintenance key process areas (KPAs) are unique to maintainers and not part of the software development function (see Table 2).

Some Maintenance Key Process Areas	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
Establishment of Service Level Agreements (SLA)	P	A
Planning of maintenance activities (versions, SLA, impact analysis)	P	A
Managing events and service requests	P	A
Supporting daily operations	P	A
Rejuvenating software	P	A

Table 2: Software management key process areas (P = present, A = absent)

When the KPAs of table 2 are compared to 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 [*Zit96, 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.

The absence in the CMM of some of the specific processes used by the maintainers in everyday situations had also been documented earlier in [*Zit96*] and they are still absent from the new CMMi© version, since it maintains a developer's view of the software production process.

3. Software Maintenance Interfaces

The revised ***SM***^{CMM} model provides a more comprehensive and more detailed maintainer's context, key interfaces and generic processes of the maintenance environment, as illustrated in the context diagram of Figure 1. There are indeed multiple interfaces in a typical software maintenance organizational context: "A 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 and keep the user-community confidence 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 mo dification 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 operations and infrastructure organization [*Iti01a, Iti01b*]. The user is rarely aware of, or involved in, internal information technology (IT) processes. Internally IT must have an effective problem resolution process and efficient communications.

A specific request, sometimes called a "ticket" when this process is automated, will typically circulate among IT 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.

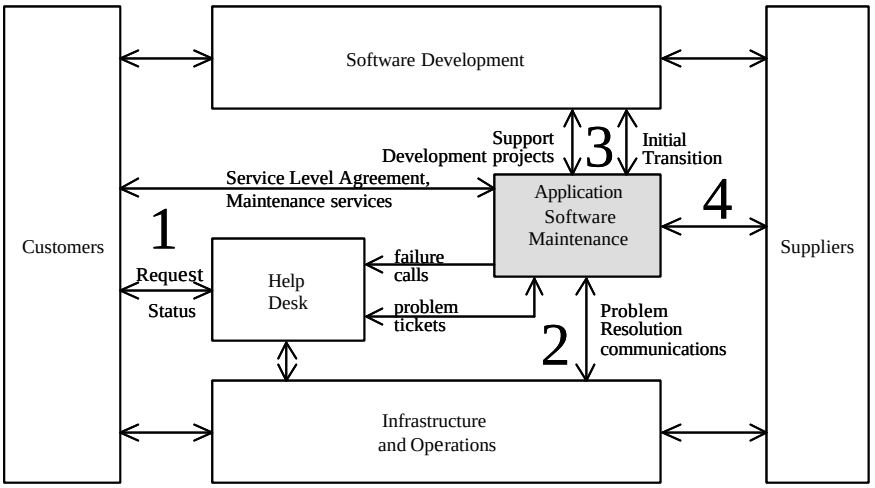

Figure 1: Software Maintainers Context Diagram

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 (figure 1) 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 replace, partially or completely, a function of the IT 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.

4. *SM*^{CMM} high level process model

To address the concerns specific to each maintenance request source, distinct maintenance process areas are required. For good alignment of the ***SM***^{CMM} with the ISO 12207 [*Iso95*] standard, the software maintenance key processes have been grouped into three classes (figure 2) [*Apr04a*]: a) primary processes (operational); b) the support processes (to the primary processes) and c) organisational processes that are offered by the IT unit or other departments of the organization (e.g. human resources, purchasing, etc.). Depending on the source of the maintenance requests, maintenance activities are handled through distinct processes; this is illustrated in Table 3 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.

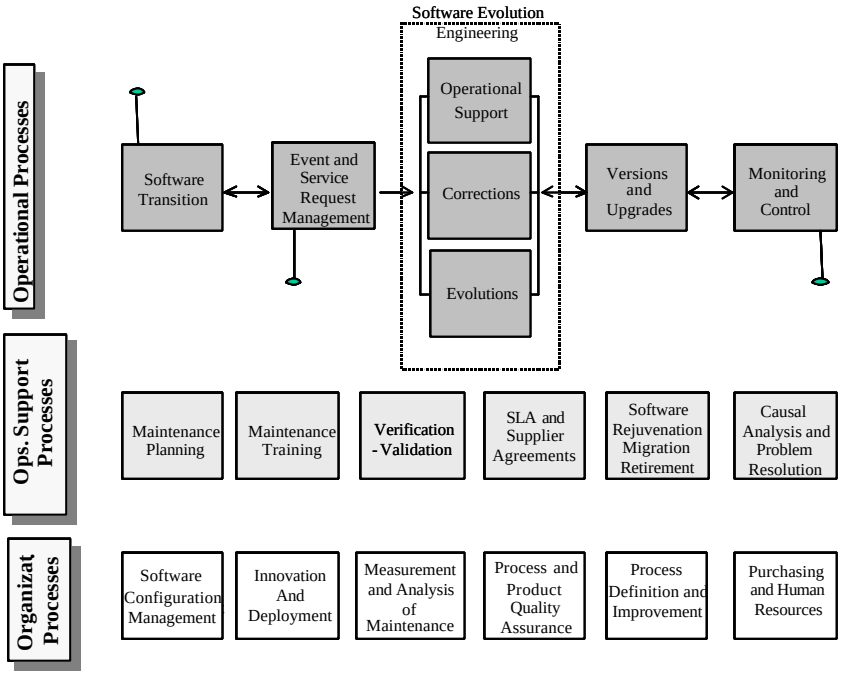

Figure 2: A classification of the Software Maintainer Key Processes – ***SM*^{CMM}** design

Source of Requests	Example of a Key Maintenance Service/Process	Assignment to a Maintenance Category of service for 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 + Service Level Agreement
Software Operations	Change to a systems utility	Perfective
Rejuvenating Studies	Software impact analysis	Often leads to a project or re-development, all of which are outside the scope of daily maintenance

Table 3: Activities and Categories of maintenance work

5. *SM*^{CMM}: Model and architecture

This section presents the model and the architecture of version 2 of ***SM***$^{CMM.}$.

5.1 Model

Version 2 of the ***SM*^CMM** is presented Table 6 (**a** and **b**) in more detail in and includes 4 Process Domains, 18 KPAs, 74 Roadmaps 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. The capability level definitions and the corresponding generic process attributes are described for each maturity level of the ***SM*^CMM** and presented in Table 4.

[*Apr04*] describes how, over a two-year period, participating organizations contributed to the mapping of each relevant practice to a capability level in the ***SM*^CMM** model.

Level– Level Name	Capability Level Definition	Process Generic Attributes
0-Inexistent	***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- Initial	***Improvised: Recognition that the practice is done 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 appear in another software maintenance section. There is no evidence that the attributes of the processes are systematically executed and that the activities are repeatable.	a) The organization is aware of the need to do 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) We cannot recognize precisely the inputs and outputs of the activity or process are; e) The is no measure of the activity or process; f) The deliverables (outputs) are not used, not easily usable and not kept up to date. Their impact is minimal; g) Who performs the activity or the qualifications/training required cannot be identified.
2- Repeatable but intuitive	***Awareness of the practice, which is either 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 sections. What characterize 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 sections.	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- Defined 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, communicated and that the employees have received proper training. We expect the qualitative characteristics of the practice or process be predictable.	a) The practice or process suggested by the model is executed; b) The same practice is used across software maintenance sections; c) Basic measures have been defined and are collected, validated 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-Managed and Measurable	***The practice is formally executed and quantitatively managed according to specified goals within established boundaries.*** Level 4 has an important distinction in 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 technique, or a similar technique, that is well suited to control the execution of the process and its most important activities. We are trying to predict the performance and control 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 take decisions to adjust or correct the causes of the deviation.
5– Optimized	***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 measurable 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) Costs and benefits studies are carried out for all innovations and major improvements; h) Activities of reuse of human resource knowledge are done; i) Causes of failure and defects (on overall activities/processes and technologies) are studied and eliminated.

Table 4: Process characteristics by process level

6. Example of a key process area – Management of Service Requests and Events

At the detailed level for each KPA, maintenance goals and key practices have been identified based on the literature on software maintenance. This section presents, as an example, a detailed description of one of the 18 KPA of the SM^{CMM}: 'Management of Service Requests and Events'. The corresponding labels for this KPA are listed in Table 5, on the basis of SPICE requirements for labeling identification.

Identifier	Key Process Area	Spice Type
Req1	Management of Service Requests and Events	2 (ORG.2)

Table 5: Example of a KPA header

6.1 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).

6.2 Objectives and goals

To ensure that the agreed-upon service levels are met, the objectives of this KPA are: a) to ensure that events and service requests are identified and registered daily; b) to determine the relative importance, within the current workload, of new events and service requests; and c) 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). 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 this KPA are as follows:

Goal_1 To proactively collect, and register all requests for services (customer-related, or internally generated);

Goal_2 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;

Goal_3 To develop a consensus on the priorities of service requests (in the queue or being processed);

Goal_4 To ensure that maintainers are working on the right (and agreed-upon) user priorities;

Goal_5 To be flexible and have the ability to interrupt the work in progress based on new events or changed priorities;

Goal_6 To proactively communicate the status of the service, planned resolution times, and current workload.

For complete operability, this KPA requires practices from other KPAs of the ***SM***^{CMM} model. As an example, linkages are required to: *Impact Analysis, Service level Agreement, Operational Support and Causal Analysis & Problem Resolution.*

Once this KPA 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.

6.3 Detailed practices

The individual practices are assigned to one of five levels of maturity. Examples of detailed practices are presented next, by maturity levels, from 0 to 3.

6.3.1 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 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 maturity level, have typically have 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.

6.3.2 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 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 is 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 of the request.

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.

6.3.3 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 IT 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.

7. Summary and next steps

This paper has presented version 2 of a software maintenance model (***SM***^{CMM}) developed to assess and improve the quality of the software maintenance function. This ***SM***^{CMM} model is based on the model developed by the SEI of the Carnegie Mellon University of Pittsburgh to evaluate and improve the process of software development. The identification of key differences between the development and the maintenance function was based on industry experience, international standards and the literature on software maintenance.

While the initial version of the model was based on only two seminal references ([*Swa89*] and [*Ball90*]), version 2 of ***SM***^{CMM} is much more broadly based and has been field-tested in two software maintenance organizations. In addition, the information provided by the ***SM***^{CMM} has been instrumental in the review and improvement of the Maintenance knowledge area for the 2004 edition of the SWEBOK Guide [*Apr03*].

Further field study is required to fine tune this maintenance 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.

The motivation for version 2 of this ***SM***^{CMM} 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***^{CMM} as a tool for continuous improvements in maintenance management could contribute to developing a better understanding of the problems of the software maintenance function.

Acknowledgments

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. 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. The opinions expressed in this paper are solely those of the authors.

8. References

[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.

[Abr04] A. Abran, J.W. Moore (Exec. Eds), P. Bourque, R. Dupuis (Eds), *Guide to the Software Engineering Body of Knowledge – 2004 Version*, IEEE Computer Society, Los Alamos, 2004. Can be downloaded free of charge from www.swebok.org

[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.

[Apr03] A. April, A. Abran, P. Bourque, "Analysis of the knowledge content and classification in the SWEBOK chapter: Software Maintenance", position paper accepted at the 11th International Workshop on Software Technology and Engineering Practice - STEP 2003, Amsterdam, Sept. 2003.

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

[Apr04a] A.April, A.Abran, R.Dumke, "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, 2004.

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

[Ben00] K.H.Bennett, (2000)c "Software Maintenance: A Tutorial", Software Engineering edited by Dorfman and Thayer, IEEE Computer Society Press, Los Alamitos.

[Bev00] Bevans N. (2000)c *Introduction to Usability Maturity Assessment*, Serco Ltd. [On line].

www.usability.serco.com (link tested on 11 November 2002).

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

[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).

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

[Cob00] IT Govenance Institute, (2000), "CobiT, Governance, Control and Audit for Information and Related Technology", 3rd Edition, July issue.

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

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

[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.

[Esi98] European Software Institute (ESI), "TeleSpice & R-Spice", [On line]. www.esi.es (Link tested on 15 November 2002).

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

[ISO95] ISO/IEC 12207, (1995). Information Technology – Software Life Cycle Processes, International Organization for Standardization, Geneva.

[ISO98] ISO/IEC FIS 14764, (1998). Software Engineering-Software Maintenance, International Organization for Standardization, Geneva.

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

[Iso00] ISO9001:2000, (2000). Quality Management Systems – Requirements, International Organization for Standardization, Third edition December 15, International Organization for Standardization, Geneva, Switzerland.

[Iti01a] Information technology Infrastructure Library, (2001), Central Computer and telecommunications Agency, Service Support, 09/2000, HSMO Books, London, UK.

[Iti01b] Information technology Infrastructure Library, (2001), Central Computer and telecommunications Agency, Service Delivery 04/2001, HSMO Books, London, UK.

[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)c "Corrective Maintenance Maturity Model: Maintainer's Education and Training", ICSE - International Conference on Software Engineering, IEEE Computer Society Press: Los Alamitos, CA.

[Ker02] Kerzner, H. (2002)c "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.

[Lud00] Ludescher, G., Usrey, M.W. (2000)c "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).

[Mal03] Malcolm Baldrige National Quality Program. (2003), "Criteria for Performance Excellence", [On line] http://www.quality.nist.gov/ (Link tested on January 8th 2003).

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

[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 30 October 2003).

[Nie02] F. Niessink, V. Clerk, H van Vliet, (2002)c "*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).

[Pig97] T.M. Pigoski. (1997), "Practical Software Maintenance: Best Practice for Managing your Software Investment", John Wiley & Sons.

[Pom02] Projxsoft, "*Project Organization Maturity Model* (POM2)", [On line]. http://www.projxsoft.com/default.asp?nc=2053&id=4 (Link tested on 11 October 2003).

[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 09 October 2003).

[Ray01] Rayner, P., Reiss, G. (2001), "*The Programme Management Maturity Model"*, The Programme Management Group, 15th February, [On line]. http://www.e-programme.com/events/pmmm.htm (Link tested on 15 October 2003).

[Sch01] Scheuing, A.Q., Fruhauf, K. (2000), "Maturity Model for IT Oprations (MITO)", [On line]. www.software.saq.ch , (Link tested on 16 December 2002).

[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 11 October 2003).

[Sch99] Schmietendorf, A., Scholz, A. (1999), "The Performance Engineering Maturity Model at a glance", Metrics News, 4(2), December issue, Magdeburg (Germany).

[Sei93] SEI (1993), "Software CMM, Version 1.1", CMU/SEI-93-TR-24, ESC-TR-93-177, Software Engineering Institute, Carnegie Mellon University, Pittsburg.

[Sei02] SEI (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, Pittsburg.

[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/cgi-bin/inetcgi/jsp/displayArticle.do?oid=15213 (Link tested on 10 October 2003).

[Str00] Stratton, R.W. (2000), 2The Earned Value Management Maturity Model2, [On line]. http://www.mgmt-technologies.com/evmtech.html (Link tested on 8 October 2003).

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

[Tob01] Tobia, E., Glynn, M. (2001), "e-business, can we control it? , e-business Maturity Model", 14th Utility Coal Conference, PricewaterhouseCoopers, [On line]. www.utilitycoalconference.com (Link tested on 10 November 2002).

[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).

[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 8 October 2003).

[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

[Zit96] M.Zitouni, A. Abran. (1996)c "A Model to Evaluate and Improve the Quality of the Software Maintenance Process", 6th International Conference on Software Quality Conference. Ottawa: ASQ-Software Division.

Process Domain	Key Process Area	Facet
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
Maintenance Request 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 (PR's and MR's plans)
	Monitoring and Control of Service Requests and Events	Follow-up on planned and approved activities
		Review and analyze progress
		Urgent changes and corrective measures
	Service Level Agreements and Supplier Agreements	Account Management of users
		Establish SLA's and contracts
		Execute services in SLA's and contracts
		Report, explain and bill services

Table 6a: SM^{CMM} Model content (Version 2)

Process Domain	Key Process Area	Facet
Software Evolution Engineering	Software Transition	Developer and Owner involvement and communications
		Transition process surveillance and management
		Training and knowledge transfer surveillance
		Transition preparation (documents, software and problem log)
		Participation in system and acceptance tests
	Operational Support	Production software monitoring
		Outside normal hours support
		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
	Software Configuration Management	Change Management
		Baseline configuration
		Reservation, follow-up and control of components and documents
	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 programme
		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	Re-documentation of software
		Restructuration of software
		Reverse engineering of software
		Re-engineering of software
		Software migration
		Software retirement

Table 6b: SM^{CMM} Model content (Version 2)