A Software Maintenance Maturity Model (*S3^m*): Measurement Practices at maturity Level 3 and 4

(Alain April[1], Alain Abran[1])

[1]École de Technologie Supérieure, Montréal, Canada, alain.april@etsmtl.ca , alain.abran@etsmtl.ca

ABSTRACT

Evaluation and continuous improvement of software maintenance are key contributors to improve software quality. The software maintenance function suffers from a scarcity of the management models that would facilitate its evaluation, management and continuous improvement. This paper presents an overview the measurements practices that are being introduced for level 3 and higher to the software maintenance maturity model (S3^m).

Key words: *Software Maintenance, Maturity Model, Process Improvement, Process Assesment*

1. INTRODUCTION

Software maintenance still does not receive a noticeable share of management attention and it is suffering from lack of planning, as illustrated often by its crisis management style; within this context, maintenance is typically perceived as being expensive and ineffective. Maintenance still suffers from a scarcity of proposals of best practices that can readily be applied in the industry. By improving the maintenance process the software engineering community expect that the product quality will also be enhanced. Researchers tend to develop isolated and very technical solutions that are quite challenging to apply in industry, more so in small maintenance groups that very small budgets and time available to improve their maintenance contexts. Alternatively, a number of software maintenance best practices implemented in some of the best run maintenance organizations still need to be recognized and better described for technology transfer to the industry at large.

For the software development function, there already exist many maturity models for evaluating the development process and for proposing improvements. For the software maintenance function, there is a much less comprehensive assessment model; this paper is based on the first comprehensive model that takes into account the characteristics specific to the maintenance process, the Software Maintenance Maturity Model - ***S3^m***. This ***S3^m*** model has been published in 'Software Maintenance Management: Evaluation and Continuous Improvement' [Apr08], including recommended practices from levels 0, 1 and 2. This paper presents now an overview of some of the measurement practices for higher levels of maturity, such as level 3 and up.

This paper is organized as follows. Section 2 introduces the many interfaces and key processes of software maintenance. Section 3 presents the measurement topics of software maintenance using sources of information for identifying the quantitative aspects necessary for level 3 and higher practices. Section 4 presents the individual measurement practices for maturity level 3 and higher. Finally section 5 and presents a summary and acknowledgments.

2. THE INTERFACES AND KEY PROCESSES OF SOFTWARE MAINTENANCE

It is important to understand the scope of maintenance activities and the context in which software maintainers' work on a daily basis (see Figure 1). There are indeed multiple interfaces in a typical software maintenance organizational context:

- Customers and users of software maintenance (labelled 1);
- Infrastructure and Operations department (labelled 2);

- Developers (labelled 3);
- Suppliers (labelled 4);
- Upfront maintenance and help desk (labelled 5).

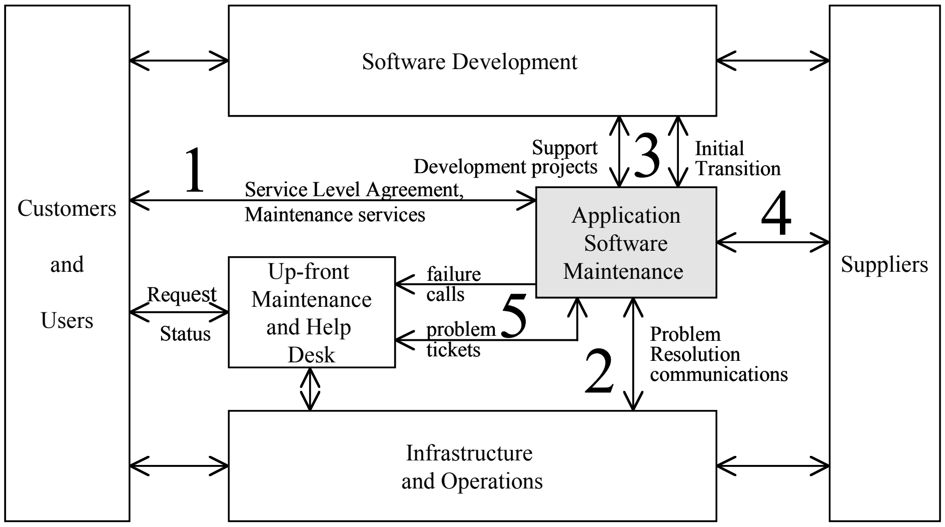
Software Development
Customers and Users
Suppliers
1
Service Level Agreement, Maintenance services
Support Development projects
3
Initial Transition
Application Software Maintenance
4
Request
Status
Up-front Maintenance and Help Desk
failure calls
problem tickets
5
2
Problem Resolution communications
Infrastructure and Operations

Figure 1: Software Maintainers Context Diagram

Taking into account that these interfaces necessitate daily services the maintenance manager must keep the applications running smoothly, react quickly to restore service when there are production problems, meet or exceed the agreed-upon level of service, 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.

Key characteristics in the nature and handling of small maintenance request have been highlighted in [Abr93], for example:

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

In the ***S3^m*** Model, the software maintenance processes are grouped into three classes (Figure 2) to provide a representation similar to that used by the ISO/IEC 12207 standard but with a focus on software maintenance processes and activities:

- Primary processes (software maintenance operational processes);
- Support processes (supporting the primary processes);
- Organizational processes offered by the Information Systems (IS) or other departments of the organization (for example: training, finance, human resources, purchasing, etc.).

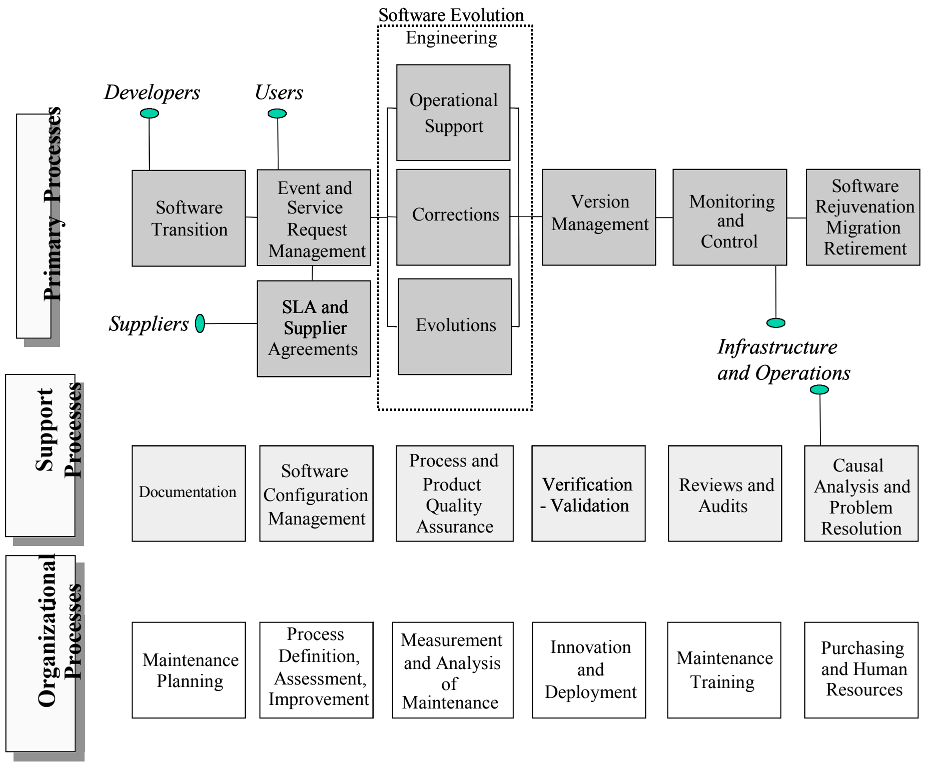
Software Evolution Engineering
Developers
Users
Primary Processes
Software Transition
Event and Service Request Management
Operational Support
Corrections
Evolutions
Version Management
Monitoring and Control
Software Rejuvenation Migration Retirement
Suppliers
SLA and Supplier Agreements
Infrastructure and Operations
Support Processes
Documentation
Software Configuration Management
Process and Product Quality Assurance
Verification - Validation
Reviews and Audits
Causal Analysis and Problem Resolution
Organizational Processes
Maintenance Planning
Process Definition, Assessment, Improvement
Measurement and Analysis of Maintenance
Innovation and Deployment
Maintenance Training
Purchasing and Human Resources

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

This generic software maintenance process model helps explain and represent the various key software maintenance 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*. This process is not limited, as is the case with some standards, to the moment when developers hand over the system to maintainers, but rather 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, and 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 *Event and Service Request Management process* handles all the daily issues, Problem Reports, Modification 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 SLA and the nature of the request and its size)[Apr01]. Supplier agreements is concerned with the management of contractual aspects (i.e. escrow, licenses, third-party) and SLAs.

Accepted requests are documented, prioritized, 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 of the software. In the model, these are referred to as "operational support" activities, and these consist of: a) replies to questions; b) provision of information and counselling; and c) helping customers to better understand the software, a transaction, or its documentation. The next primary processes concern the *Version Management process* that moves items to production, and the *Monitoring and Control process*, ensuring that the operational environment has not been degraded. Maintainers always monitor the behavior of the operational system and its environments for signs of degradation. They will quickly warn other support groups (operators, technical support, scheduling, networks, and desktop support) when something unusual happens and judge whether or not an instance of service degradation has occurred that needs to be investigated. The last primary process addresses rejuvenation activities to improve maintainability, migration activities to move a system to another environment and retirement activities when a system is decommissioned.

A process which is used, when required, by an operational process is said to be an operational support process. In most organizations both the developers and the maintainers share these processes. In this classification, we include: a) the documentation process; b) the software configuration management function and tools which are often shared with developers; c) the process and product quality assurance; d) the verification and validation processes; e) the reviews and audits processes; and, finally, f) the problem resolution process, that is 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 (e.g. the many maintenance planning perspectives, process related activities, measurement, innovation, training, and human resources). While it is important to measure and assess these processes, it is more important for the maintainer to define and optimize the operational processes first. The operational support processes and the organizational processes follow these.

3.OVERVIEW OF INFORMATION SOURCES FOR SOFTWARE MAINTENANCE MEASUREMENT PRACTICES OF PROCESS, PRODUCT AND SERVICE.

The following section identifies information sources used by the S3M for measuring the software maintenance processes, products and services.

3.1 SOFTWARE MAINTENANCE PROCESS MEASUREMENT INFORMATION SOURCES

A process is defined as a sequence of steps performed for a given purpose. It is observed that the quality of software is largely determined by the quality of the development process used to design it [Pau95]. The maintenance manager's objective, then, is to help bring such a process under control, and measurement has an important role to play in helping him meet this objective. Because he has little control over the development of the software, the maintainer must identify the earliest point at which he can influence the maintainability characteristics of the new software under construction. Initiating measurement during pre-delivery and transition could be a key strategy to assess the quality of the product and its readiness to be accepted in maintenance. To achieve this goal, a decision can be made to implement a software maintenance measurement program and link it to the software development project measurements. For example, if the maintainer can set some maintainability objectives early on in the development of new software, its quality could be measured both during and after development.

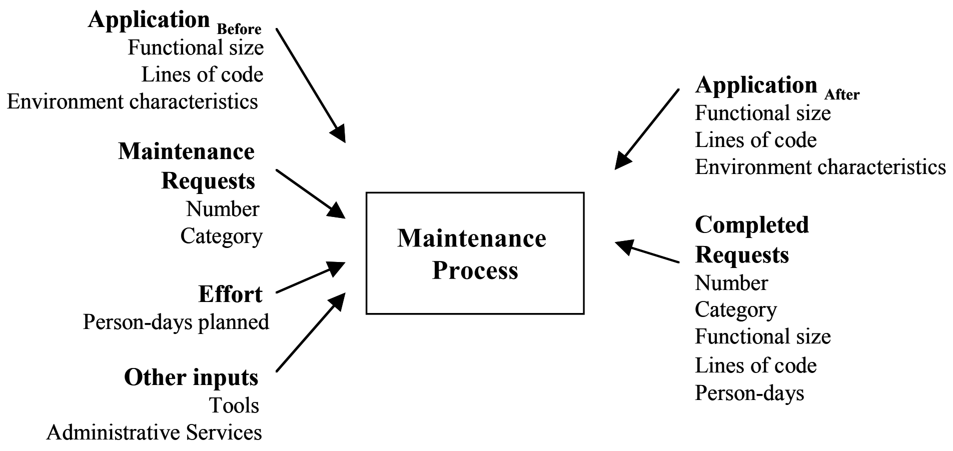

Figure 3: Example of selected characteristics of a maintenance process [Des97]

Many factors must be taken into account before measuring software maintenance processes [Pfl90]. One strategy is to identify the key activities of a the maintenance process. These key activities, in turn, have characteristics that can be measured. But before measures can be identified it is essential that the software maintenance processes be defined first. Software maintenance measurement pre-requisites have been presented in April and Al-Shurougi [Apr00], that identified 5 pre-measurement requirements in terms of: 1) definitions of work categories; 2) implementation of request management process/software; 3) classification of maintenance effort in an activity account data chart (billable/unbillable); 4) implementation of activity management (time sheet) software, data verification and, 5) the measurement of the size of requests for change.

The SEI [SEI02] describes process measurement activities as appearing at level 2 of maturity. This confirms the need for a defined process before measurement can be initiated. While the SEI recommended measures could have been a starting point for maintainers, Pigoski [Pig97] noted that the measures proposed by the SEI were created from a development perspective and do not capture the peculiarities of software maintenance. Other authors [McG95, Des97] also confirm this view and specify that a software maintenance measurement program must be planned separately from that of the developers. The measurement requirements being different, software maintenance measures are more focused on problem resolution and on the management of change requests.

Higher maturity organizations have established a maintenance measurement program. For instance, Grady and Caswell [Gra87] identify the following concerns that are specific to a software maintenance measurement program:

- “How should we plan for staffing maintenance of a software product family?
- When is a product stable?
- Is “mean time between failures (MTBF)” for a software product really a meaningful measure of software quality?
- In which development phase should tool and training investments be made?
- How does documentation improve supportability?
- How many defects can be expected in a project of a given size?
- What relationships exist between defects found prior to release and those found after release?
- What, if any, is the relationship between the size of a product and the average time to fix a defect?
- What is the average time to fix a defect?
- How much testing is necessary to ensure that a fix is correct?
- What percentage of defects are introduced during maintenance? During enhancements?”

3.2 SOFTWARE MAINTENANCE PRODUCT MEASUREMENT INFORMATION SOURCES

Two different perspectives of maintainability are often presented in software engineering literature [ISO9126]. From an external point of view, maintainability attempts to measure the effort required to diagnose, analyze and apply a change to specific application software. From an internal product point of view, the idea is to measure the attributes of application software that influence the effort required to modify it. The internal measure of maintainability is not direct, meaning that there is no single measure for the application's software maintainability and that it is necessary to measure many sub-characteristics in order to draw conclusions about it [Pre01].

The IEEE1061 standard [IEE98] also provides examples of measures without prescribing any of them in particular. By adopting the point of view that reliability is an important characteristic of software quality, the IEEE 982.2 guide [IEE88] proposes a dictionary of measures.

3.3 SOFTWARE MAINTENANCE SERVICE MEASUREMENT INFORMATION SOURCES

Some authors propose that "software maintenance has more service-like aspects than software development" [Nie00]. This seems to be the case for other IS/IT services as well, like computer operations. Service quality measures for software maintenance services have been proposed in the literature. They are divided into two categories:

A) Service agreements (a Service Level Agreement, a service contract and outsourcing contract)
B) Software maintenance benchmarking

A) SERVICE LEVEL AGREEMENT

To reach agreement on service levels, a consensus must be developed between the customers and the maintenance organization about the related concepts, terms and measures to be used. Service Level Agreements (SLAs) are said to be internal when they are exercised entirely within a single organization. This type of agreement documents consensus about activities/results and targets of the many maintenance services. SLAs originally appeared in large computer operations centres during the 1950s [McB90]. They progressively extended their reach to all IS/IT service-oriented activities during the 1970s. However, many IS/IT organizations still do not have SLAs in place today [Kar08].

Some attempts to identify exemplary practices and propose SLA maturity models have appeared recently. CobiT is a set of exemplary practices used by IS/IT auditors. CobiT identifies and describes practices that must be implemented in order to meet the IS/IT auditor requirements for SLAs. It identifies the "definition and management of the service level" as an essential practice, with the measurement of quality of service as its main objective. CobiT describes five maturity levels for the agreement (nonexistent, ad hoc, repeatable, defined, managed/measured and optimised) [COB07]. CobiT also describes other software engineering processes that are directly related to software maintenance. Other proposals of SLA maturity models have recently been put forward can be found in [Nie05, Kaj04].

SLAs become an important element of customer satisfaction in a competitive environment. A few publications have directly addressed SLAs in a software maintenance context [Mue94, Bou99, Nie00, Apr01, Kaj04, Nie05, COB07].

Additionally, the SLA should clarify the expectations/requirements of each service. In the context of an internal agreement on software maintenance services between the maintenance organization and their users/customers, two attitudes have been reported [Apr01]:

1) On the one hand, the customer wants to concentrate on his business and expects a homogeneous service from his IS/IT organization. The result of this homogeneous service for the customer is the ability to work with a set of information systems without any disturbance whatever from the source of the failure. The way in which this service, these systems or their infrastructure are constituted is of less interest to the customer, who would like this vision to be reflected in his SLA. This means that a result-based SLA on the total service (including help desk and computer

operations) should be described, and not only on the individual/partial IS/IT components (i.e. a server, a network, software).

2) On the other hand, software maintainers only own a portion of the service as perceived by the customer. They are often quite ready to provide a result-based SLA for their services. But the products are operating on infrastructures that are not under their responsibility (desktop, networks and platforms), as they only interface with the help desk and computer operations. To achieve integrated measurement of all the components of the software implies the involvement of all groups concerned, and implies that someone in IS/IT must own an overall SLA where all the IS/IT services are described.

In [Apr01], the internal SLA of a Cable & Wireless member company is described in detail. [Apr01] also concurs with the opinion of other authors, which is that all the IS/IT service organizations that support the customer must be documented in a unified SLA to satisfy the customer. A unified SLA is one that includes the service levels of all the IS/IT organizations that are involved in supporting the end users.

B) SOFTWARE MAINTENANCE BENCHMARKING

A popular definition of benchmarking was originally reported by David T. Kearns, CEO of Xerox: 'Benchmarking is the continuous process of measuring products, services and practices against the toughest competitors or those companies recognised as industry leaders'. Benchmarking's prerequisite is a clear understanding of internal processes. While it's not necessary to have a very elaborate measurement program, there must be data on hand. Some argue that benchmarking could be a waste of time if the organization does not possess that understanding and a certain number of internal process measurements.

Organizations which use this approach specifically for maintenance, present, at the end of the exercise, variants of the following graphs/measures:

- Function Points supported per person (in house development)
- Function Points supported per person (in house development + software packages)
- Cost by supported function points
- Average age (in years) of application software (currently in production)
- Age groups (% by functionality) of application software (currently in production)
- Number of supported programming languages
- Supported data structures (Sequential, Indexed, Relational, Others)
- % of programming types (Maintenance, Improvements, New applications)
- Support – Environment complexity index (based on the number of users, data size, platform size and power)
- Support – Technical diversity (number of applications, programming languages, data archiving technologies, tools and operating systems)
- Support – Use of CASE tools
- % of personnel stability (turnover rate)
- Duration of employment (Years)
- Human resources level (% of total company employees)
- Salaries
- Training effort (number of training days per person per year)
- Rate of faults in production (by criticality : critical, major, minor or cosmetic) per 1000 supported FPs
- Rate of faults in production (by cause category : design, programming, environment or other) per 1000 supported FPs

Benchmarking activities with open databases is now feasible but requires sustained help from a knowledgeable champion. Organizations in other, often more mature, industries publish insightful results from the use of this benchmarking practice.

4. MAINTENANCE PROCESS AND PRODUCT PERFORMANCE ADVANCED PRACTICES

We have presented the main sources that were used to develop the measurement practices of this maturity model. At maturity level 3 (see table 1) process, product and service measures should be defined and implemented. Implementing measurement should be considered as part of an organizational process improvement project in the organization. The software maintenance measures should also be co-located with the software development and operations measures. Key activities and services of the institutionalized software maintenance processes have to be identified as candidates to be measured. Quality attributes of applications software maintained need also to be identified as candidates to be measured. Measures objectives (targets) and baselines (current value) are then documented. These objectives should be consistent with business objectives and the specific context of each organization. After measures have been well defined there is a need to identify their sources, data gathering activities and validation needs. Maintainers are trained on measurement and validation activities. Data is collected, as needed, by maintainers at the operational level. Data is incorporated into organizational repositories where it can be used to develop measurement models. Customer reporting of maintenance measure follows. At this point in time the customers, of software maintenance, should perceive IT measures harmonization across activities, services and applications software.

PRACTICE	DESCRIPTION
Pro4.3.1	Managers of software maintenance identify risks that may cause measurement failure of products, intermediate products and application software maintained.
Pro4.3.2	Each application software maintained has a candidate set of quality and service measures.
Pro4.3.3	The software maintenance organization identifies processes and key activities of software maintenance and identifies its quality measures.
Pro4.3.4	The software maintenance organization sets, for each selected measure, a quality objective and a performance target.
Pro4.3.5	Candidates' analytical techniques, to be used for statistical analysis of software maintenance measurements are identified and assessed.
Pro4.3.6	Each software maintenance process, application software maintained and product has a benchmark for analysis, control and follow-up their progress over time.
Pro4.3.7	Process performance models are designed.
Pro4.3.8	Software perfomance models are designed.

Table 1: S3M measurement practices at maturity level 3

At maturity level four (see table 2), process of establishing measures of processes and products as well as its key activities are characterized by its quantitative aspects that are entered in order to manage objectives achievement. At this level of maturity, impact of processes and software quality are statistically proven and measures already in place are optimized. Attributes of performance quality and applications software are optimised and deepened. Intervals of objectives and baselines are statistically demonstrated. Mechanized analysis of software allow definition of additional measures to try to cover all the four sub-features of 'maintainability'. Customer perceives a measures harmonization of activities, services and applications software. Measurements of internal programming standards are included in compilers, assemblers, links, operating systems loaders, testing tools and documentation tools. Models for predicting failures evolve to be used to decide whether a change can be promoted into production.

PRACTICE	DESCRIPTION
Pro4.4.1	Estimated data are stored in a database to be used to improve process and to plan future requests (size estimation of request, data on required work, on productivity, on defects).
Pro4.4.2	Maintenance organizational units measure their productivity in a quantitative way.
Pro4.4.3	Internal measures of software maintainability are the subject to extending, tools and quantitative management. They are explained in simple terms to all stakeholders

Pro4.4.4	Establish and maintain an understanding of gap in performance on the use of key activities, selected measures and analytical techniques.

Table 2: S3M measurement practices at maturity level 4

At maturity level five models to manage software maintenance demand and product reliability are typically used to assess resources or schedules using measures from past events/requests (size, duration, effort, criticality of defect). Results of measurement analysis (on maintenance processes and products) are also used to monitor the processes and its critical sub-processes (under statistical control).

5. SUMMARY

This paper has presented the software maintenance measurement publications that have been used to allocate process and product measurement practices in a maturity model. Further research work is required to experiment how the model helps software maintenance organizations in their improvement activities. For example, it would be most appropriate and useful to analyze the results of empirical studies in order to demonstrate clearly the proposed location of each of the practices. This would ensure that key practices suggested by maintenance experts or described in the literature are can be used in industry and yield the promised benefits. Empirical studies could also be set up to study the efficiency of the model as a tool for continuous improvements in maintenance management. The empirical studies would contribute to a better understanding of the problems of the software maintenance function and in the validation of the proposed model.

6. AKNOWLEDGEMENTS

We thank industry members contributing to this project and offer special thanks to Laxman Vasan of IBM GSA Australia and Caroline Milam of Freescale USA. This research was carried out at the Software Engineering Research Laboratory at the École de Technologie Supérieure – University of Quebec, headed by Dr. Alain Abran. The opinions expressed in this report are solely those of the authors.

7. REFERENCES

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

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

[Apr00] April, A.; Al-Shurougi, D.: Software Product Measurement for supplier Evaluation, Proceeding of the Software Measurement Conference (FESMA-AEMES), Madrid, Spain, October 18-20, 2000, Http://www.lrgl.uqam.ca/publications/pdf/583.pdf [February 10, 2008].

[Apr08] April, A, Abran, A (2008). Software Maintenance management: Evaluation and Continuous Improvement, IEEE Computer Society, isbn: 9780470147078, 320p.

[Bou99] Bouman, J.; Trienekens, J.; Van der Zwan, M. (1999). Specification of Service Level Agreements, Clarifying Concepts on the Basis of Practical Research, Proceedings of Software Technology and Engineering Practice '99.

[COB07] IT Governance Institute, (2007). CobiT, Governance, Control and Audit for Information and Related Technology, Version 4.1.

[Des97] Desharnais, J-M.; Paré, F.; St-Pierre, D. (1997). Implementing a Measurement Program in Software Maintenance – an Experience Report Based on Basili's Approach, presented at the IFPUG Conference, Cincinnati, OH.

[Gra87] Grady, R.; Caswell, D. (1987). Software Metrics: Establishing a Company-wide Program. Englewood Cliffs, NJ, Prentice-Hall.

[IEE98] Institute of Electrical and Electronics Engineers. IEEE Standard for a Software Quality Metrics Methodology, Standard 1061-1998. Institute of Electrical and Electronics Engineers: New York, NY, 1998; 35 p.
[IEE88] Institute of Electrical and Electronics Engineers. IEEE Guide for the use of the IEEE Standard Dictionary of measures to produce reliable software, Standard 982.2-1988: 96p.
[Kaj04] Kajko-Mattsson M.; Ahnlund, C.; Lundberg, E. (2004). CM3: Service Level Agreement, Proceedings of the IEEE International Conference on Software Maintenance (ICSM 2004). Chicago, IL, USA, 432—436.
[Kar08] Karten, N.: Establishing Service Level Agreements. Karten Associates. Http://www.nkarten.com/slaservices.html (Link tested on February 10, 2008).
[Lag96] Laguë, B.; April, A. (1996). Mapping of the ISO9126 Maintainability Internal Metrics to an industrial research tool, Proceedings of SES 1996, Montreal, October 21-25, Http://www.lrgl.uqam.ca/sponsored/ses96/paper/lague.html (Link tested on February 10, 2008).
[McB90] McBride, D. (1990). Service Level Agreements: A Path to Effective System Management and Application Optimization, Hewlett-Packard Professional.
[McG95] McGarry, J. (1995). Practical Software Measurement: A Guide to Objective Program Insight, Department of Defense, September 1995.
[Mue94] Mueller, B. (1994). Software maintenance agreements poised for change, Systems Management Journal.
[Nie05] Niessink, F.; Clerk, V.; van Vliet, H.: The IT service capability maturity model, 1.0 release candidate 1, Software Engineering Research Centre: Utrecht, The Netherlands, 2005; 224 p. Http://www.itservicecmm.org/ (Link tested on February 10, 2008).
[Nie00] Niessink, F.: Perspectives on Improving Software Maintenance, SIKS P.H.D dissertation 2000-1, Dutch Graduate School for Information and Knowledge Systems.
[Pig97] Pigoski, T.M. (1997). Practical Software Maintenance: Best Practice for Managing your Software Investment, Wiley.
[Pau95] Paulk, M. (1995). The evolution of the SEI's capability maturity model for software. Software Process 1, August 1995; 50—60.
[Pfl90] Pfleeger, S.L.; Bohner, S. (1990). A framework for maintenance metrics. Proceedings of the Conference on Software Maintenance (Orlando, FL), IEEE Computer Society Press.
[Pre01] Pressman, R.S.: Software Engineering: A Practitioner's Approach. McGraw-Hill, New York, NY, 2001; 860 p.
[SEI02] Capability Maturity Model Integration for Software Engineering (CMMi), (2002). Version 1.1, CMU/SEI-2002-TR-028, ESC-TR-2002-028, Software Engineering Institute, Carnegie Mellon University.