Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Electronic Notes in Theoretical Computer Science 233 (2009) 73–87
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

A Software Maintenance Maturity Model (*S3M*): Measurement Practices at Maturity Levels 3 and 4

Alain April and Alain Abran

École de Technologie Supérieure, Montréal, Canada
Email: alain.apriletsmtl.ca, alain.abranetsmtl.ca

Abstract

Evaluation and continuous improvement of software maintenance are key contributors to improving software quality. The software maintenance function suffers from a scarcity of the management models that would facilitate these functions. This paper presents an overview of the measurement practices that are being introduced for level 3 and higher to the Software Maintenance Maturity Model (***S3M***).

Keywords: Software Maintenance, Maturity Model, Process Improvement, Process Assessment, Product Assessment

1 Introduction

Software maintenance still does not receive a noticeable share of management attention and suffers from lack of planning, as often illustrated by its crisis management style. Part of the problem is that maintenance is typically perceived as being expensive and ineffective. Moreover, few proposals of best practices have been put forward which can readily be applied in industry. In general, the software engineering community expects that product quality will be enhanced if the maintenance process is improved. However, researchers tend to develop isolated and very technical solutions which are quite challenging to apply in industry, all the more so in small maintenance groups which have very small budgets and limited time available to improve their maintenance activities. At the same time, a number of software maintenance best practices have been implemented in some of the best run maintenance organizations, although these practices have yet to be recognized and need to be better described to prepare them for technology transfer to industry at large.

For the software development function, many maturity models already exist for evaluating the development process and for proposing improvements. For the soft-

1571-0661/
doi:10.1016/j.entcs.2009.02.062

ware maintenance function, the assessment models available are much less comprehensive. This paper is based on the first comprehensive model to take into account the characteristics specific to the maintenance process, the Software Maintenance Maturity Model – ***S3M***. This model has recently been published in 'Software Maintenance Management: Evaluation and Continuous Improvement' [2], including recommended software measurement practices at levels 0, 1 and 2 [4]. This paper presents an overview of some of the advanced measurement practices identified for higher levels of maturity, such as level 3 and up. The model references elements from other model such as ISO/IEC-12207, ISO/IEC-9126, CMMI, and IEEE-1061. An overview of the model describing explicitly the nature and the elements of each maturity level is available in [3]. Finally it would be beneficial if the authors could provide a case study with the application of the proposed model and the results of its use.

The 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 practices at maturity level 3 and higher. Section 4 presents the advanced measurement practices at maturity level 3 and higher. Finally, section 5 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 more detailed explanation of this figure can be found in [3]). There are indeed multiple interfaces in a typical software maintenance organizational context:

- Customers and users of software maintenance (labeled 1 in Figure 1);
- Infrastructure and operations department (labeled 2);
- Developers (labeled 3);
- Suppliers (labeled 4);

- Up-front maintenance and help desk (labeled 5).

Taking into account that these interfaces necessitate daily service, the maintenance manager must keep the applications running smoothly, react quickly to restore service when there are production problems, meet or exceed the agreed level of service, maintain the confidence of the user community that they have a dedicated and competent support team at their disposal which is acting within the agreed budget.

Key characteristics in the nature and handling of small maintenance requests have been highlighted in [1], for example:

- Modification requests come in more or less randomly and cannot be accounted

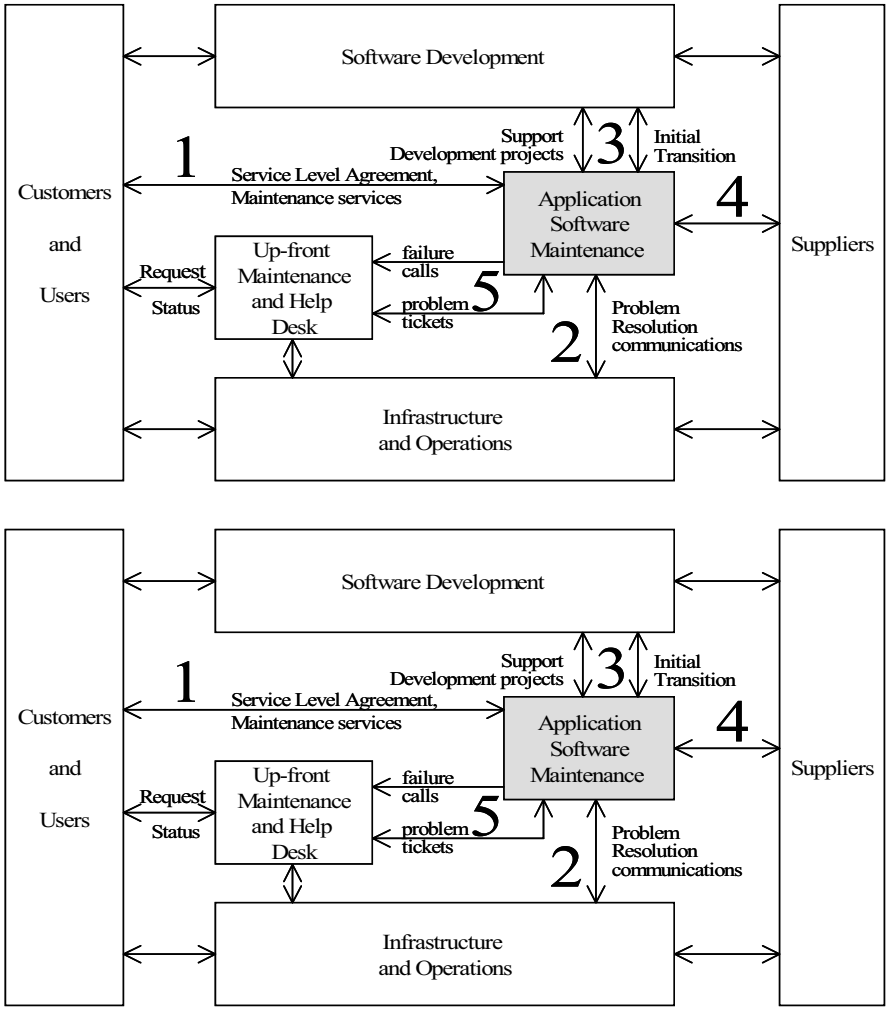

Fig. 1. Diagrammatic representation of the software maintainer context

for individually in the annual budget-planning process;

- Modification requests are reviewed and assigned priorities, often at the operational level, and most of them do not require senior management involvement;
- The maintenance workload is managed using queue management rather than project 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 a request for a correction can take priority over other work in progress;

In the ***S3M***, the software maintenance processes are grouped into three classes (Figure 2) to provide a representation similar to that used by the ISO 12207 standard, but with a focus on software maintenance processes and activities:

(i) Primary processes (software maintenance operational processes);

(ii) Support processes (supporting the primary processes);

(iii) Organizational processes offered by Information Systems (IS) or other departments of the organization (for example, Training, Finance, Human Resources, Purchasing, etc.).

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 *Software Transition* process. This process is not limited 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 or not a request is to be addressed, rerouted, or rejected (on the basis of the Service Level Agreement (SLA) and the nature and size of the request) [5]. The *SLA and Supplier Agreements* process is concerned with the management of contractual aspects (e.g. escrow, licenses, a 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. These are referred to as operational support activities in the model, and they consist of: a) replies to questions; b) provision of information and counselling; and c) helping customers better understand the software, a transaction, or its documentation. The next primary processes concern the *Version Management* process, which moves items to production, and the *Monitoring and Control* process, which ensures 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, *Rejuvenation, Migration and Retirement,* addresses activities to improve maintainability, migration activities to move a system to another environment, and

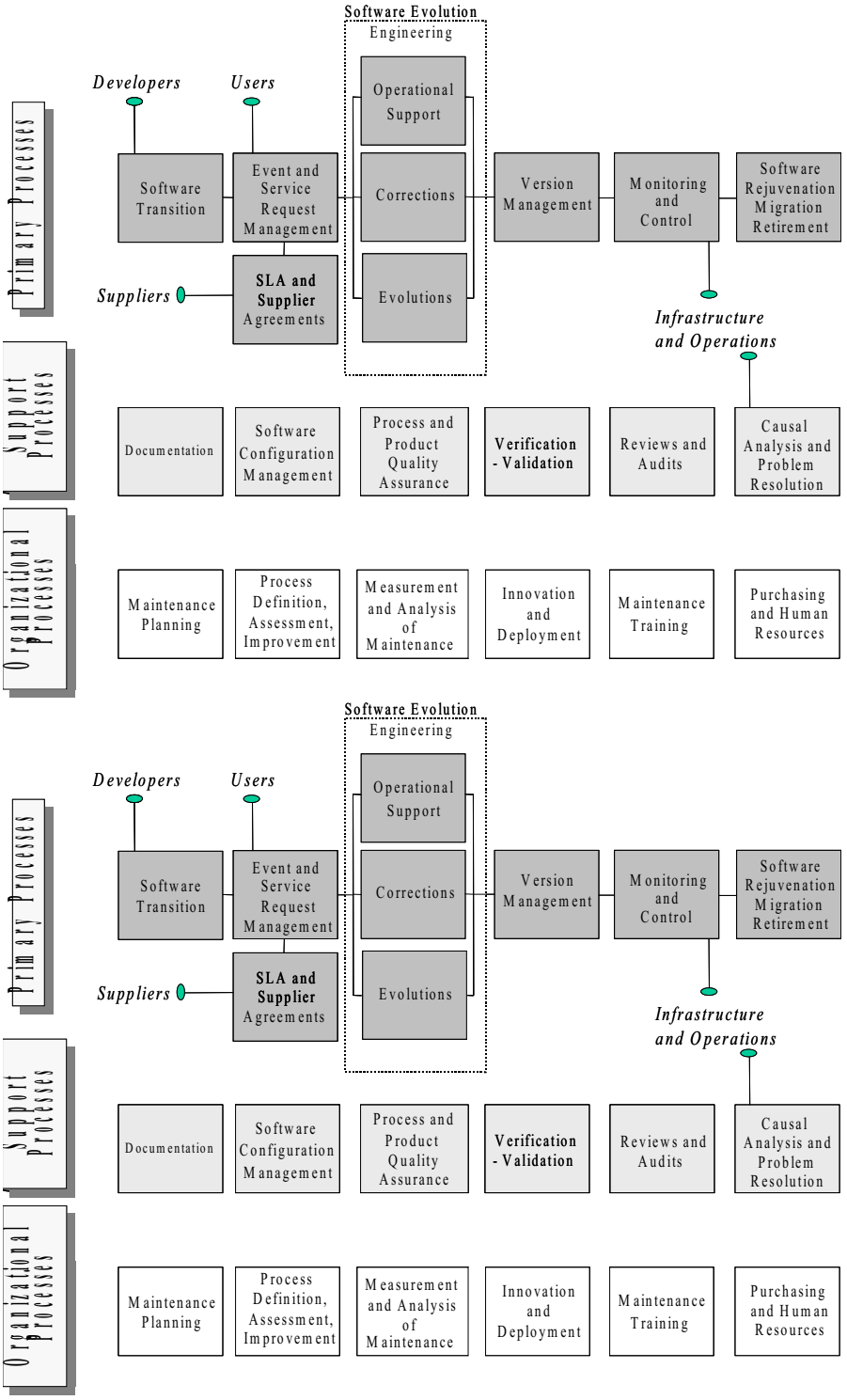

Fig. 2. A classification of the software maintainer's Key Processes.

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, these processes are shared by both the developers and the maintainers. This classification includes: a) the documentation process; b) the software configuration management function and tools, which are often shared with developers; c) process and product quality assurance; d) the verification and validation processes; e) the review and audit processes; and, finally, f) the problem resolution process, which is often shared with the infrastructure and operations departments. These are all key processes which are required to support software operational maintenance 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 Information SOURCES for MAINTENANCE Measurement Practices

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

3.1 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 [22] and to maintain it. 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 the maintenance manager has little control over the development of the software, he 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 is a key strategy in assessing the quality of the product being developed and its readiness to be accepted in maintenance. To achieve this objective, a decision can be made to implement a maintenance measurement program and link it to the software development project measurements. For example, if the maintainer can set some maintainability targets early on in the development of new software, its quality could be measured both during and after development.

Many factors must be taken into account before measuring software maintenance processes [23]. One strategy is to identify the key activities of the process. These key activities have characteristics, which can be measured, but, before measures can be identified, it is essential that the software maintenance processes be defined. Software maintenance measurement prerequisites were presented in April

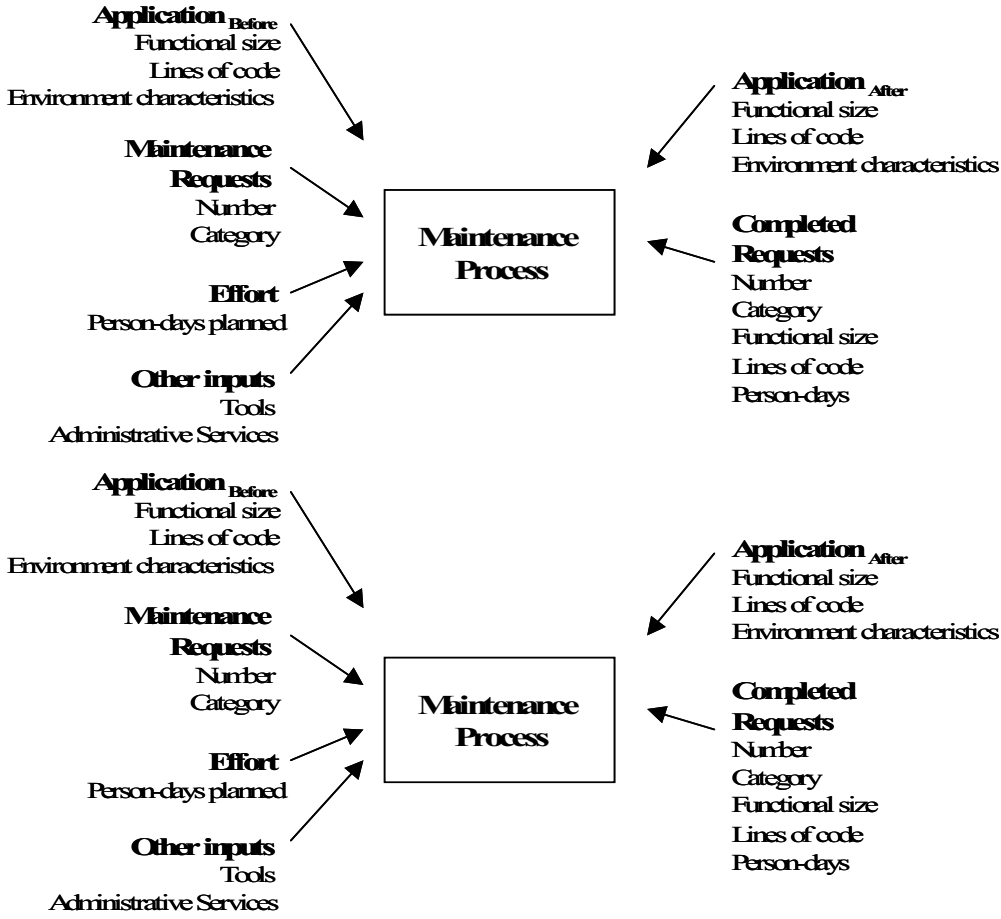

Fig. 3. Example of selected characteristics of a maintenance process [Des97]

and Al-Shurougi [6]: 1) definition of maintenance work categories; 2) implementation of a process for requests management; 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) measurement of the size of change requests.

The SEI [25] describes process measurement activities as appearing at maturity level 2. This confirms the need for a defined process before measurement can be initiated. While the SEI's recommended measures could have been a starting point for maintainers, Pigoski [21] noted that those measures were created from a development perspective and do not capture the unique features of software maintenance. Other authors [17,9] confirm this view, and specify that a software maintenance measurement program must be planned separately from that of the developers: because the measurement requirements are 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 [10] identify the following concerns that are specific to a software maintenance measurement program:

How should we plan for staffing the maintenance of a software product family?

When is a product stable?

Is the 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 is introduced during maintenance? During enhancements?

3.2 Maintenance Product - Measurement INFORMATION SOURCES

Two different perspectives of maintainability are often presented in the software engineering literature [15]. 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 of the application's maintainability and that it is necessary to measure many sub-characteristics in order to draw conclusions about it [24].

The IEEE 1061 standard [11] 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 [12] proposes a dictionary of measures.

3.3 Maintenance Service - Measurement INFORMATION SOURCES

Some authors believe that "software maintenance has more service-like aspects than software development" [20]. 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 an outsourcing contract)

B) Software maintenance benchmarking

A) Service Level Agreement (SLA)

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 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 centers during the 1950s [16]. 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 [14].

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: it identifies and describes practices that must be implemented in order to meet the IS/IT auditor requirements for SLAs. It also 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 optimized) [8]. CobiT also describes software engineering processes which are directly related to software maintenance. Other proposals of SLA maturity models can be found in [19,13].

SLAs become an important element of customer satisfaction in a competitive environment. A few publications have directly addressed these in a software maintenance context [18,7,20,5,13,19,8].

The SLA should clarify the expectations/requirements of each service. In the context of an internal agreement on software maintenance services between a maintenance organization and their users/customers, two attitudes have been reported [5]:

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 results-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 results-based SLA for their services. But the products are operating on infrastructures which are not their responsibility (desktop, networks, 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 that someone in IS/IT must own an overall SLA in which all the IS/IT services are described.

In [5], a case study of an internal SLA is described in detail. [5] also concurs with the opinion of other authors, in 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 which includes the service levels of all the IS/IT organizations 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 recognized as industry leaders." Benchmarking's prerequisite is a clear understanding of internal processes. While it is not necessary for the measurement program to be very elaborate, there must be data on hand.

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

(i) Function Points (FP) supported per person (in-house development)
(ii) Function Points (FP) supported per person (in-house development + software packages)
(iii) Cost based on supported function points
(iv) Average age (in years) of application software (currently in production)
(v) Age groups (% by functionality) of application software (currently in production)
(vi) Number of supported programming languages
(vii) Supported data structures (Sequential, Indexed, Relational, other)
(viii) % of programming types (Maintenance, Improvements, New applications)
(ix) Support – Environment complexity index (based on the number of users, data size, platform size, and power)
(x) Support – Technical diversity (number of applications, programming languages, data archiving technologies, tools, operating systems)
(xi) Support – Use of CASE tools
(xii) % of personnel stability (Turnover)
(xiii) Duration of employment (in years)
(xiv) Human resources level (% of total number of employees)
(xv) Salaries
(xvi) Training effort (number of training days per person per year)
(xvii) Rate of faults in production (by criticality: critical, major, minor, or cosmetic) per 1000 supported FP's
(xviii) Rate of faults in production (by cause category: design, programming, environment, or other) per 1000 supported FP's

The benchmarking of activities with open databases is now feasible, such as the one from the International Software Benchmarking Standards Group – IS-

BSG (www.isbsg.org). Benchmarking 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 Measurement Advanced Practices

We have presented in Section 3 the main sources used to identify the advanced measurement practices of this maturity model. At maturity level 3 measures of process, product, and service have been defined and implemented: implementing measurement is 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 been identified as candidates to be measured. The quality attributes of the application software maintained also need to be identified as candidates to be measured. Measurement objectives (targets) and baselines (current value) are documented: these targets are consistent with the business objectives and specific context of each organization.

After measures have been well defined, there is a need to identify their sources, data gathering activities, and verification requirements. Maintainers are trained on measurement and verification activities. Data are collected, as needed, by maintainers at the operational level, and then incorporated into organizational repositories where they can be used to develop measurement models for maintenance purposes. This is followed by customer reporting of maintenance measures. At this level 3 of process maturity, there should be a perception on the part of software maintenance customers of harmonization of the IT measures across activities, services, and application software.

Table 1 presents a set of eight advanced measurement practices that can be observed at maturity level 3. For instance, practice Pro4.3.2 'All maintained application software have a candidate set of quality and service measures': for each application software, there has been an analysis of which of the ISO 9126 quality characteristics and sub-characteristics are relevant, with which relative weights, and measured with which specific base and derived measures. For each ones, a specific target has been set and data collected is collected to monitor progress against this target. Similarly, a service level agreement has been defined for each application, and progress is tracked against the service levels specified in the corresponding maintenance contract. Details for each of these advanced measurement practices can be found in the references identified in Section 3.

At maturity level 4 (see Table 2), the process of establishing measures of processes and products, as well as its key activities, is characterized by its quantitative aspects, which are designed in order to manage achievement of the objectives. At this level of maturity, the impact of processes and software quality is statistically demonstrated and measures already in place are optimized. Attributes of performance quality and application software are reviewed and analyzed. Intervals of objectives and baselines are statistically determined. Mechanized analysis of soft-

Practice	description
Pro4.3.1	Managers of software maintenance identify measures to report cause of failure (of process, intermediate products, and production software).
Pro4.3.2	All maintained application software have a candidate set of quality and service measures.
Pro4.3.3	The software maintenance organization identifies processes and key activities of software maintenance and their related quality measures.
Pro4.3.4	The software maintenance organization sets a quality objective and a performance target for each selected measure.
Pro4.3.5	Candidate analytical techniques for statistical analysis of software maintenance measurements results are identified and assessed.
Pro4.3.6	Every software maintenance process, maintained application software, and product has a baseline for analysis, control and follow-up of their progress over time.
Pro4.3.7	Process performance models are designed, implemented and monitored
Pro4.3.8	Software performance models are designed, implemented and monitored.

Table 1
S3M measurement practices at maturity level 3

ware allows for the definition of additional measures to increase coverage of all four sub-characteristics of maintainability. The customer perceives a harmonization of measures of activities, services, and application software. Measurements of internal programming standards are included in compilers, assemblers, links, operating system loaders, testing tools, and documentation tools. Models for predicting failures evolve to be used to decide whether or not a change can be promoted to production.

Table 2 presents a set of four advanced measurement practices that can be observed at maturity level 4. For instance, practice Pro4.4.2 'Maintenance organization units measure their productivity in a quantitative way': for each application software, effort data is collected for each request for changes to software functional requirements , each such change is sized using an ISO recognized measurement method for functional sizing, and productivity ratios are calculated using relevant additional application characteristics. The distribution of maintenance requests across maintenance categories are monitored and analyzed to allow meaningful comparisons with past data and to make adjustments to maintenance estimation models for effort by maintenance categories. Details for each of these advanced measure-

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, data on productivity, data on defects).
Pro4.4.2	Maintenance organization units measure their productivity in a quantitative way.
Pro4.4.3	Internal measures of software maintainability are subject to further definition, automation in tools, and quantitative management. They are explained in simple terms to all stakeholders.
Pro4.4.4	An understanding of gaps in performance on the use of key activities, selected measures, and analytical techniques is established and maintained.

Table 2
S3M measurement practices at maturity level 4

ment practices can be found in the references identified in Section 3.

At maturity level 5, models for managing 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 defects). Results of measurement analysis (on maintenance processes and products) are also used to monitor the processes and its critical sub-processes to ensure that they are managed under statistical process control.

5 Summary

This paper has presented the software maintenance measurement information sources that have been used to identify process and product measurement practices in the ***S3M*** maturity model for software maintenance. Two sets of advanced measurement practices have been presented for maturity levels 3 and 4. Further work is required to determine how the model helps software maintenance organizations in their improvement activities. For example, it would be highly appropriate and useful to analyze the results of empirical studies in order to demonstrate clearly the proposed location of each of the practices. For example the practices of Table 1 and Table 2 should be analytically explained for better understanding. This would ensure that key practices suggested by maintenance experts or described in the literature 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 improvement 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.

Acknowledgement

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 Quèbec, headed by Dr. Alain Abran. The opinions expressed in this report are solely those of the authors.

References

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

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

[3] A. April, J. Huffman Hayes, A. Abran, R. Dumke, "Software Maintenance Maturity Model (SMmm): The software maintenance process model", JSME: Research and Practice, 17 May/June 2005:197-223.

[4] A. April, A. Abran, R. Dumke, "Software Maintenance Productivity Measurement: How to assess the readiness of your organization", Proceedings of the International Workshop on Software Metrics and DASMA Software Metrik congress, IWSM/MetriKon 2004, Shaker-Verlag, Knigs Westerhausen, Germany, 2004:231-244.

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

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

[7] 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.

[8] IT Governance Institute, (2007). CobiT, Version 4.1.

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

[10] Grady, R., Caswell, D. (1987). Software Metrics. Englewood Cliffs, NJ, Prentice-Hall.

[11] 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.

[12] 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.

[13] 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.

[14] Karten, N. Establishing Service Level Agreements. Karten Associates. http://www.nkarten.com/slaservices.html

[15] Lagu, 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

[16] McBride, D. (1990). Service Level Agreements: A Path to Effective System Management and Application Optimization, Hewlett-Packard Professional.

[17] McGarry, J. (1995). Practical Software Measurement: A Guide to Objective Program Insight, Department of Defense, September 1995.

[18] Mueller, B. (1994). Software maintenance agreements poised for change, Systems Management Journal.

[19] 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/

[20] Niessink, F. Perspectives on Improving Software Maintenance, SIKS Ph.D dissertation 2000-1, Dutch Graduate School for Information and Knowledge Systems.

[21] Pigoski, T.M. (1997). Practical Software Maintenance: Best Practice for Managing your Software Investment, Wiley.

[22] Paulk, M. (1995). The evolution of the SEI's SW-CMM Software Process 1, August 1995; 50-60.

[23] Pfleeger, S.L., Bohner, S. (1990). A framework for maintenance metrics. Proceedings of the Conference on Software Maintenance (Orlando, FL), IEEE Computer Society Press.

[24] Pressman, R.S. Software Engineering: A Practitioner's Approach. McGraw-Hill, New York, NY, 2001; 860p.

[25] 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.