Assessment Results using the Software Maintenance Maturity Model (*S^3m^*)

David-Alexandre Paquette Alain April Alain Abran

École de Technologie Supérieure
david-alexandre.paquette.1@ens.etsmtl.ca
alain.april@etsmtl.ca
alain.abran@etsmtl.ca

Abstract

*This **S^{3m}** maintenance maturity assessment model is divided into four process domains containing 18 "Key Process Area", each in turn containing "Roadmaps". Roadmaps are bodies of knowledge containing recommended practices that are linked to one another. Using the **S^{3m}** software maintenance maturity model, this paper describes the assessment process and results of an individual maintainer process maintaining a key software application within a larger software maintenance organization.*

1. Introduction

In an organization, a large amount of resources is spent on maintenance to keep software operational. Unfortunately, software maintenance is often neglected by management, and this lack of planning frequently results in crisis management [April 2004, 2005]. In fact, it has been reported that "the biggest problem in software maintenance is not technical but rather its management" [Colter 87]. Furthermore, organizations are now adopting less substantial development methodologies, which leads to incomplete documentation being turned over to maintainers, which in turn puts maintenance at risk and makes it prone to high maintenance costs.

Notwithstanding these difficulties, assessing a software maintenance in an organization is a management responsibility, and it is preferable that such an assessment be performed using a recognized assessment model that is specifically relevant to maintenance. Assessment of the maturity level of an organization's processes provides a diagnosis of the current situation by comparing it with the reference model at the core of the assessment methodology. Such a diagnosis then leads to the proposal of an improvement plan or corrective actions in a specific area in which weaknesses have been identified.

The Software Maintenance Maturity Model – **S^{3m}** [April 2005] – proposes a process model to assess the maturity of the software maintenance practices in an organization. The **S^{3m}** model includes all the unique software maintenance activities not included in the CMMi [SEI 2000], and its proposed processes and activities can be used to design improvements to the software maintenance process. With this model, it is possible to determine the level of maturity of a maintenance organization and how to improve its practices.

This paper presents a case study illustrating a maintenance maturity assessment using the **S^{3m}** model. It is organized as follows: section 2 presents an overview of the **S^{3m}** maintenance maturity model and section 3 an overview of the organization being assessed, while section 4 presents the assessment procedure adopted and section 5 the assessment results. A summary and recommendations are presented in section 6.

2. Maintenance Maturity Model

Poor knowledge of best practices and their benefits to software maintenance activities can lead to inefficient or inappropriate investment. This in turn: a) creates dissatisfaction on the client's part; b) hinders minimization of delays and service costs; and c) delays the deployment of improvements, innovation and investment in information technology.

Organizations can use the $\boldsymbol{S^{3m}}$ model to launch and sustain a continuous improvement program tailored to software maintenance by initially benchmarking their current maintenance practices against the model. This will help maintenance organizations identify their strengths and weaknesses in delivering software maintenance services. With the identification of a gap, maintenance organizations can identify, through a comparison with the model, what issues to address and how to address them, and by doing so improve their maintenance processes.

The scope of a model such as $\boldsymbol{S^{3m}}$ is to deal with the software maintenance processes that are under an organization's direct control. A practical approach was used to apply proven knowledge in software maintenance engineering to offer relevant practices to improve maintenance, and to do so for all types of industries and for organizations of any size.

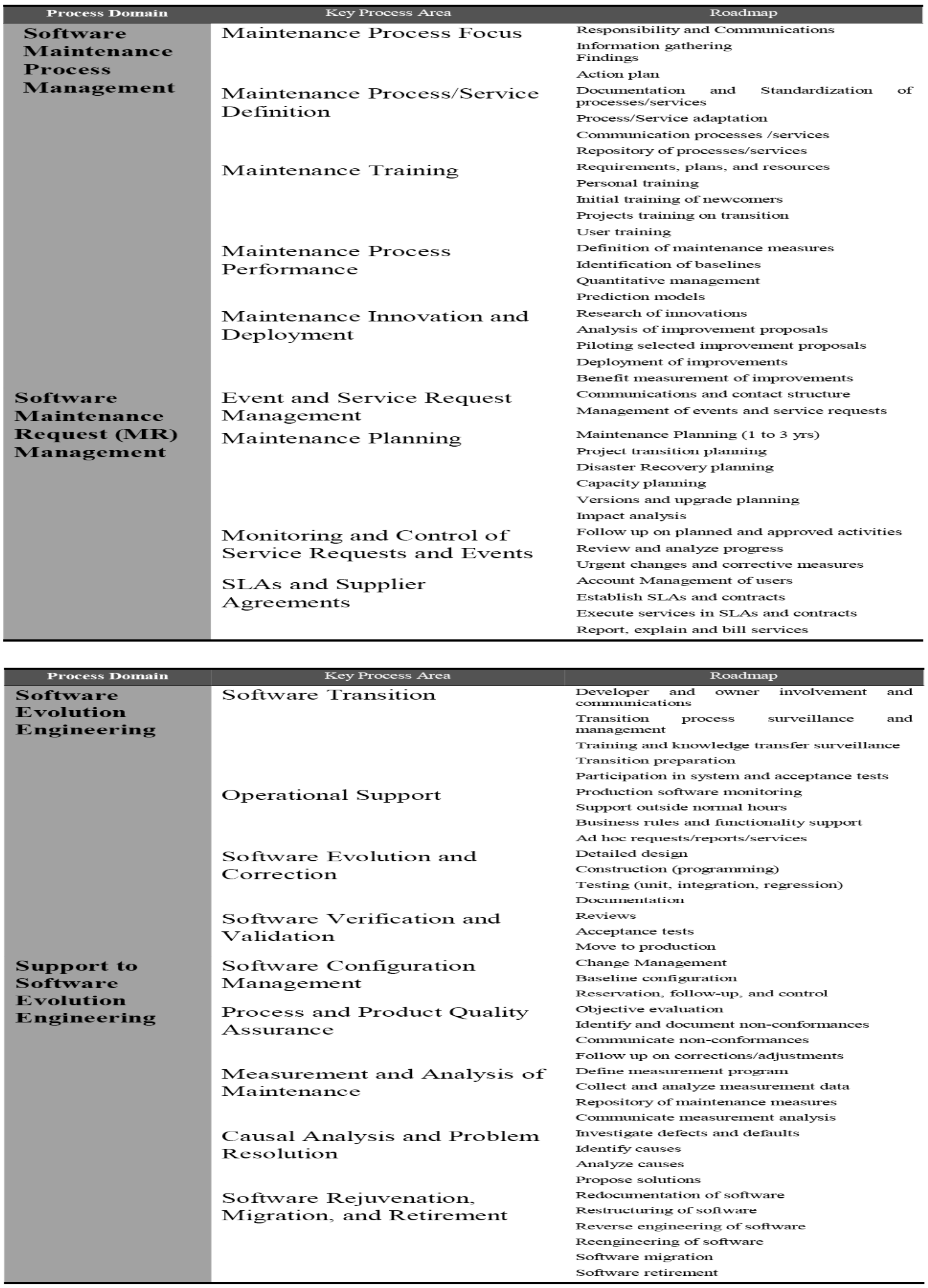
It is an organization's responsibility to use this model for meeting their business objectives, while taking into consideration the organization's costs and constraints.

The scope of $\boldsymbol{S^{3m}}$ is limited to small maintenance activities, and the model is not appropriate for software maintenance projects of larger scope which should be managed as projects using project management techniques. For the maintenance workload requiring project management expertise and techniques, CMMi and other maturity models should be used.

The $\boldsymbol{S^{3m}}$ model:

- is based on the customers' perspective;
- is relevant for the maintenance of application software: a) developed and maintained in-house; b) configured and maintained in-house or with a subcontractor's help; and c) outsourced to a supplier;
- provides references and details for each best practice;
- offers an improvement approach based on roadmaps and maintenance categories;
- covers international software life cycle processes and maintenance standards like ISO 12207, ISO 14764, ISO 9001, ISO 20000 and ISO 14764;
- covers relevant parts of the CMMi, a reference model for software improvement.

The $\boldsymbol{S^{3m}}$ includes four process domains: 1 - Maintenance process management; 2 - Maintenance request management; 3 - Software evolution engineering; and 4 - Support to software evolution engineering, as well as 18 Key Process Areas – KPAs – and 74 Roadmaps for maintenance practices – see Figure 1.



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

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

Figure 1: S^{3m} structure

The ***S**^{3m}* model integrates references to additional software maintenance practices documented in other popular software and quality improvement models. Its intention is to facilitate maintenance assessment and improvement activities by concentrating all pertinent practices in a single integrated model.

3. The Assessed Organization

The assessed organization offers IT services to a single large customer. A large portion of IT services support and maintain well-known software packages which have been adapted to, and implemented in, the customer's operations. Over the years, this organization has carried out less and less in-house development, and increasingly more support and maintenance. It has an ISO-9001:2000-certified quality system, which includes the IT services within its scope.

One of the software products the organization maintains is an invoicing process. This software application invoices the customers for their use of software, computer hardware and telephone equipment. Recently, the software has experienced billing inaccuracies. The internal audit department asked for a verification of the accuracy of the invoicing, and requested a comparison of the results of a manual tally against the software system's invoice content. This audit highlighted a number of inaccuracies, mainly in the component inventory, including defects which sometimes generated losses for the organization, along with occasional customer over-billing.

A project was launched to address this issue. However, there was no plan in place for conducting it. Not only that, but a single staff resource was deemed to be sufficient to conduct the impact analysis. This individual was the only person available at that time with experience in the programming language and database management system. Once the impact analysis had been completed, it was decided that additional functionality would be developed to address the data accuracy problem. To save time, and under management pressure to do this quickly, no documentation was produced, and maintenance and support, whenever necessary, was carried out by the single resource assigned to the project. This new software has a number of users: the foremen, those in charge of the various inventories (software, hardware and telephone equipment) and one individual in the accounting department. The first version of the software helped monitor inventory changes and detect problems prior to billing. Errors found in this version were progressively corrected, and users identified various potential enhancements. The second version added functionality, within the same project constraints in terms of time pressure and staff. Fortunately, documentation was written for the system in this second version, including a user guide and a small PowerPoint presentation.

Maintenance remained the responsibility of the one resource, progressively becoming a secondary task. This resource left to fulfill other commitments, at which point the maintenance duties were transferred to another maintenance programmer. Unfortunately, knowledge transfer could not take place, with the result that the users returned to the old process.

This software is now an important part of the invoicing system; however, the users object to the inadequacy of the process, which is proving very difficult to carry out. Within this context of crisis management, the new maintainer can only successfully "extinguish fires", which immediately reignite. As a result, management is beginning to recognize the importance of implementing sound maintenance practices.

4. Assessment Procedure

In some software maintenance organizations which are very small (sometimes a single maintainer), and in others where a SCAMPI-type assessment [SEI 2000] is impractical, for example, in a setting in which maintainers want feedback on their processes, but cannot devote three full-time weeks to an "official" CMMi-type assessment, an ***S**^{3m}* mini-assessment method has been developed. This method yields a reliable maturity rating without the investment of unavailable resources and also permits the selection of individual assessment components to focus the investigation on specific concerns and to tailor the scope of the assessment and rating effort to a level relevant to the individual software maintenance organization.

The S^{3m} model's practices, for maturity levels 0, 1 and 2, have been placed in an Excel-based questionnaire. In this organization's assessment, the questions/statements were addressed during meetings with the software maintenance resource and senior management. These meetings also helped in the exchange of information and to raise awareness of the model's practices proposed at each level. Once compiled, the results present the current maintenance program's strengths and weaknesses, and its corresponding maturity level. The assessment methodology rating notation closely follows the recommendations of the model's creators. Since the assessor in this case study had no previous experience with the model or the assessment method, the maturity rating procedure was simplified to improve the objectivity of the responses to each question/statement associated with the first two levels of maturity (0 and 1):

For level 0, the admissible responses to the questions/statements are “True” and “False”. All the statements are stated in the negative form to try to confirm the absence, rather than the presence, of a specific software maintenance process. Take, for example, **Req1.0.1** The software maintenance organization does not manage user requests or software events. Responding “True” to this question generates a rating of 0% and “False” a rating of 100%.

For levels 1 and 2, the choice of responses is based on the scale suggested by the authors of ***S*^{3m}** [April 2003, 2004, 2005], that is, in conformity with ISO 15504 [ISO 2004]:

N: Not Achieved, 0-15%: There is no or little evidence that the process objectives and goals have been met.
P: Partially Achieved, 16%-50%: Some of the objectives and goals of the process have been met.
L: Largely Achieved: 51%-85%: A significant portion of the objectives and goals of the process have been met.
F: Fully Achieved: 86%-100%: The objectives and goals of the process have been fully met.

To facilitate the calculation of the percentages and to reduce possible subjectivity because of a lack of experience of the assessor, the value 0% was assigned if the process was not carried out, or “Not Achieved”. For the other rating levels (P, L and F), the median value was used, giving the following four possible ratings:

N: Not Achieved 0 %
P: Partially Achieved (50% - 16%) / 2 + 16% = 33%
L: Largely Achieved (85% - 51%) / 2 + 51% = 68%
F: Fully Achieved (100% - 86%[1]) / 2 + 86% = 93%

The organization sponsors reviewed and accepted this rating approach.

5. Assessment Results

This section presents the results of the ***S*^{3m}** maturity assessment of the maintenance of this invoicing software.

Maturity level 0 practices assessment

Level 0 is the entry point to ***S*^{3m}** and provides an overview of each Key Process Area for the four process domains. The assessment results are illustrated in Table 1. It can be observed that this maintenance organization does not consistently fulfill the criteria of each process domain. For the Process Management domain, the organization fulfills only some of the requirements: for instance, software maintenance is performed without a life cycle, and software maintenance work is perceived as a marginal activity.

Furthermore, no software training is planned or provided to the maintainer, nor is there any monitoring of maintenance process performance and no process measurement. Time sheets are collected, but they are not analyzed or used for improving software maintenance. Of course, there is no measurement for the purpose of innovation in the current software maintenance activities and technologies.

In contrast, this organization is interested in this process assessment and therefore meets the criteria of practice **5.0.2**: Study of a solution for improvement, and **5.0.3**: Impact of an improvement contained in the KPA Maintenance innovation and deployment.

With an overall rating of only 29%, this process domain is not under control, and this has a definitive impact on the quality of the service given to customers.

These findings have raised awareness at the management level that action is required in the short term.

The second process domain of the ***S*^{3m}** model, Event/Request Management, was also assessed at level 0, since no corresponding process is implemented by this organization: there is no process for managing user requests, all interactions being conducted informally between the users and the maintainers. For instance, users phone the

[1] The 86% value represents the minimum for the “Fully Achieved” rating in ISO 15504.

maintainer directly as soon as they encounter a problem with the software, and no trace of this event is kept. The maintainer does his best to handle the most urgent situations. Maintenance is performed based on the availability of resources, and, if the maintainer is on sick leave or is assigned to another task, maintenance is postponed. In contrast, if the software monthly cycle cannot be executed at the end of the month, senior management assigns it top-priority maintenance status. In summary, none of the KPAs in this domain is performed in this organization.

The third process domain, Evolution Engineering, contains the basic analysis and programming activities. This process domain of the $\boldsymbol{S^{3m}}$ model is rarely missed if some level of maintenance is performed, even if it is fairly basic. Each of the process areas is carried out in this organization. Through interviews, it was established that having a single maintainer assigned to development and maintenance helps in this situation. Transition and predelivery activities are conducted for each phase of the project by the resource responsible for the software. Impact analysis, programming and tests are validated with the users. Support is given to users after corrections and modifications have been completed. Users are also informed by email of the dates and times of failures and interruptions in the running of the software. Unfortunately, these emailed failure data are not included in monthly reports of failures and downtime. The employee assigned to maintenance tasks also bases his reports on the same fundamental processes, and stresses that there is little documentation in the source code and that the validations and internal controls are limited. This process domain meets all criteria for level 0.

In the fourth process domain, Support to Evolution Engineering, the only process area that meets the criteria is the KPA Configuration and Version Management: the maintainer has developed a folder structure to classify the source versions. This very basic configuration management technique works well when there is only one maintainer. None of the other process areas of this fourth domain is executed, since no independent quality assurance activity is conducted on the maintainer's work and there is no documented software quality guidance in the corporate quality system. The individual in charge of corporate quality assurance does not carry out any audit in this area of the organization. The analysis of the causes of failure does not follow a problem resolution process, this being left to the discretion of the maintainer. Lastly, there is no plan for rejuvenating the software under maintenance. In summary, only 20% of the criteria in this process domain are met.

The aggregated rating of the four process domains is 37%, which corresponds only to the "Partially Achieved", 16%-50% scale level for level 0 of maintenance maturity in $\boldsymbol{S^{3m}}$.

Maturity levels 1 and 2 practices assessment

Even though the level 0 scale rating is well below the 86% passing target, there is still interest in assessing practices at levels 1 and 2.

At maturity level 1, the process applied in the organization is characterized as the work of the individual maintainer who carries out the software maintenance. This characterization maps well to what is happening in this specific organization. For the first process area, some improvement work is being carried out.

The rating given in response to the directive ***Pro1.1.1 Rate how informal the improvement of software maintenance processes is*** is "Largely Achieved", because some key improvements are being made. Technical improvements have also been reported, consisting in this organization in the implementation of programming rules and file structures.

On that basis, the statement ***Pro1.1.2 Some individual improvement initiatives aim mostly at improving technical aspects of software maintenance processes*** is rated by the assessor as "Fully Achieved". The maintenance and service support processes are also based on the expertise and initiatives of the individual maintainer. Some personal notes and embryonic processes have been initiated on an ad hoc basis by the individual maintainer.

Although the individual initiatives were ad hoc, the assessor rated the questions referring to them:
Pro2.1.1 Are the maintenance processes/services informal and based on the experience of individuals? and
Pro2.1.2 Are individual initiatives, for defining maintenance processes/services, mainly trying to address technical aspects of the software or describe, in a local format, the activities of a specific maintenance organization?
and the statement:
Pro5.1.2 Individual initiatives, for improvement and innovation, target mainly the technical aspects of software maintenance
as accurately representing what is done in this organization, even though there are few formal artifacts.

This process assessment activity contributed to rating question ***Pro5.1.3 Are assessments of new processes, technologies, methodologies and tools for maintenance are performed informally?*** as "Largely Achieved" in this case.

Process Domain	Process Area	Level 0 Question	Rating	% Completed
Process management	Maintenance process focus	1.0.1	Yes	0%
	Maintenance process/service definition	2.0.1	Yes	0%
	Maintenance training	3.0.1	Yes	0%
	Maintenance process performance	4.0.1	Yes	0%
	Maintenance innovation and deployment	5.0.1	Yes	0%
		5.0.2	No	100%
		5.0.3	No	100%
Total				29%
Event/request management	Event/request management	1.0.1	Yes	0%
	Maintenance planning	2.0.1	Yes	0%
	Requests/software monitoring and control	3.0.1	Yes	0%
	SLA and supplier agreements management	4.0.1	Yes	0%
Total				0%
Evolution Engineering	Predelivery and transition services	1.0.1	No	100%
	Operational support services	2.0.1	No	100%
	Software evolution and correction services	3.0.1	No	100%
	Verification and validation	4.0.1	No	100%
Total				100%
Support to Evolution Engineering	Configuration and version management	1.0.1	No	100%
	Process, service and software quality assurance	2.0.1	Yes	0%
	Maintenance measurement and analysis	3.0.1	Yes	0%
	Causal analysis and problem resolution	4.0.1	Yes	0%
	Software rejuvenation, migration and retirement	5.0.1	Yes	0%
Total				20%
Level 0 Rating:				37%

Table 1: Summary of level 0 assessment results

The next process domain, Evolution Engineering, is again assessed as quite strong. All process areas were rated as "Fully Achieved" and with the highest rating of 93%. In this specific context, the developer is also the maintainer: transition is therefore not an issue and is awarded full marks. Support, although informal, is offered by the individual who developed the initial software: when a failure occurs, the users can phone the maintainer or walk to his desk. All maintenance services are executed using the maintainer's ad hoc processes. Verification and validation are performed informally after each change to production, the timing varying based on the availability of the maintainer. All level 1 questions for this process domain reflect the situation of this 'one person' group.

Of the KPAs of the process domain Support to Evolution Engineering, only one is executed. Again, configuration management is carried out by a single resource. This resource checks and validates his work himself, because he is the only one with the required expertise. All the other level 1 practices are rated as "Not Achieved". The overall rating for maturity level 1 is 36%, that is, "Partially achieved".

Table 2 presents the results of the practices that rated higher than "Not Achieved" for level 1 of ***S***^{3m}.

Process Domain	Process Area	Level 1 Question	Rating	% Completed
Process management	Maintenance process focus	1.1.1	L: Largely Achieved	68%
		1.1.2	F:Fully Achieved	93%
	Maintenance process/service definition	2.1.1	L: Largely Achieved	68%
		2.1.2	L: Largely Achieved	68%
	Maintenance innovation and deployment	5.1.2	L: Largely Achieved	68%
		5.1.3	L: Largely Achieved	68%
Total				36%
Evolution Engineering	Pre-delivery and transition services	1.1.1	F:Fully Achieved	93%
	Operational support services	2.1.1	F:Fully Achieved	93%
	Software evolution and correction services	3.1.1	F:Fully Achieved	93%
	Verification and validation	4.1.1	F:Fully Achieved	93%
Total				93%
Support to Evolution Engineering	Configuration and version management	1.1.1	F:Fully Achieved	93%
Total				15,5%
Level 2 Rating:				36%

Table 2 : Summary of the level 1 assessment results (first three process domains).

The assessment for maturity level 2 identified only two process areas with some processes rated higher than "Not Achieved". These two process areas fall into the Evolution Engineering process domain:
- ad hoc reporting and data extractions are performed by the maintainer and are rated "Largely Achieved";
- evolutions and corrections are made in the original programming language throughout the maintenance.

Table 3 shows that only the following two practices have been rated higher than "Not achieved" for level 2 of ***S***^{3M}.

Process Domain	Process Area	Roadmap	Level 2 Question	Rating	% Completed
Evolution Engineering	Operational support services	Ad hoc requests/reports/services	2.2.6	L: Largely Achieved	68%
	Software evolution and correction services	Evolution/Correction	3.2.5	F: Fully Achieved	93%

Table 3: Summary of level 2 assessment results (4th process domain).

5. Summary and Recommendations

In the organization being assessed, the ***S***^{3M} assessment has made it possible to better understand the current work status of the maintenance of a single software application, as well as the maintenance issues arising from some of the failures of the development process. For instance, pressures from senior management had led to a quick and functional project, but with very weak documentation by the only resource assigned to it. This resource was initially assigned to the maintenance of this software, but left some time later. After his departure, the process weaknesses became more serious and obvious. For instance, the new resource assigned to the maintenance of this software did not receive training on the software, and had only limited documentation at his disposal. Maintenance activity became hectic for software that was critical to the business operations. Under these conditions, management focus was required and the ***S^{3m}*** was used to obtain a diagnosis by evaluating the maturity level of the maintenance of this software.

The results of this evaluation showed us that this software maintenance organization did not meet the requirements of level 0 of the ***Sm^{M}*** maturity model for the invoice application. While the Evolution Engineering process domain was assessed at level 1, and a KPA at level 2, two of the other three process domains did not even receive the level 0 rating of "Largely Achieved" (i.e. within the 51% - 85% range).

Following presentation of the assessment results, several emails circulated at management level. Management is now aware of the benchmarking results with the ***S^{3m}*** model, which provided documentary evidence that their maintenance processes for this software under maintenance was creeping along at level 0 on the maturity scale! To address this issue in a satisfactory manner, various steps had to be taken.

Various recommendations were tabled before the management team to improve their maintenance, based on the ***S^{3m}*** model, which, if accepted, could constitute the first step in a plan to improve the organization's maintenance maturity level. The key to success in implementing them lies is senior management commitment and continuous support for the action items proposed, which are the following:

First, it is of primary importance to have senior management's support in order to officially recognize the importance of software maintenance within the organization. Second, an improvement project for software maintenance must be initiated and supported with appropriate budget and resources. Third, documentation and prioritization of the multiple and sometimes concurrent maintenance requests submitted to the maintainer is recommended to ensure that management is aware of, and understands, the amount of effort involved. Fourth, the maintenance manager is to prepare a list of all the possible short-term improvements that can be implemented. Fifth, improvement activities based on the ***S^{3M}*** are to be assigned to different individuals based on their expertise. For instance, process areas involving process and management are to be assigned to managers, and more technical process areas to programmers. The final action item recommended is to conduct another maturity assessment to monitor how the organization is progressing.

Assessing software maintenance using ***S^{3m}*** enabled this organization to better identify and understand their current process weaknesses and to present a roadmap for change to their management team.

References

[April 2005] April, A.; Hayes, J. Huffman; Abran, A.; Dumke, R., **Software Maintenance Maturity Model (*SM^{mm}*):The software maintenance process model**, Journal of Software Maintenance and Evolution: Research and Practice ,vol. 17(3), 2005, pp. 197-223.

[April 2004] April, A.; Abran, A.; Dumke, R., ***SM^{cmm}*** **Model to Evaluate and Improve the Quality of Software Maintenance Process: Overview of the model**, SPICE 2004 Conference on Process Assessment and Improvement, Critical Software SA, Lisbon (Portugal), The Spice User Group, 2004, pp. 19-32. Http://www.gelog.etsmtl.ca/publications/pdf/812.pdf

[April 2003] April, Alain; Abran, Alain; Reiner R, Dumke, **Software Maintenance Capability Maturity Model (*SM^{CMM}*): Process Performance Measurement**, International Workshop on Software Measurement (IWSM), Montreal, Shaker-Verlag 2003, pp. 16. Http://www.gelog.estmtl.ca/publications/pdf/781.pdf

[Colter 1987] Colter, M., **The Business of Software Maintenance**, *1st Workshop Software Maintenance,* University of Durham, Durham (England), 1987.

[ISO 2004] International Organization for Standardization. Information technology: Process Assessment – Part 4: Guidance on use for process improvement and process capability determination, ISO/IEC 15504-4:2004. International Organization for Standardization, Geneva, 2004.

[SEI 2000] Standard CMMi Appraisal Method for Process Improvement (SCAMPI) : Method Description, (2000). Version 1.0, CMU/SEI-2000-TR-009, Software Engineering Institute, Carnegie Mellon University.