Software Maintenance Maturity Model (SM^{mm}): The software maintenance process model

Alain April[1], Jane Huffman Hayes*,†,[2], Alain Abran[1], and Reiner Dumke[3]

[1]*Department of Software Engineering, Université du Québec, École de Technologie Supérieure, Canada*
[2]*Department of Computer Science, University of Kentucky, U.S.A.*
[3]*Department of Informatik, Otto von Guericke Universtity of Magdeburg, Germany.*

SUMMARY

We address the assessment and improvement of the software maintenance function by proposing improvements to the software maintenance standards and introducing a proposed maturity model for daily software maintenance activities: Software Maintenance Maturity Model (**SM^{mm}**). The software maintenance function suffers from a scarcity of management models to facilitate its evaluation, management, and continuous improvement. The **SM^{mm}** addresses the unique activities of software maintenance while preserving a structure similar to that of the CMMi©[4] maturity model. It is designed to be used as a complement to this model. The **SM^{mm}** is based on practitioners' experience, international standards, and the seminal literature on software maintenance. We present the model's purpose, scope, foundation, and architecture, followed by its initial validation.

J. Softw. Maint. And Evolution 2004;

No. of Figures: 4. No. of Tables: 7. No. of References: 117.

KEY WORDS**:** software maintenance; process improvement; process model; maturity model

*Correspondence to: Jane Hayes, Computer Science, Laboratory for Advanced Networking, University of Kentucky, 301 Rose Street, Hardymon Building, Lexington, Kentucky 40506-0495 USA.
†E-mail: hayes@cs.uky.edu

Contract/grant sponsor: none

1. INTRODUCTION

Corporations that rely on revenues from developing and maintaining software now face a new, globally competitive market with increasingly demanding customers. With services and products available from vendors the world over, customers are insisting that these services and products be of high quality, cost as little as possible, and be accompanied by support services that challenge the competition. To satisfy these needs, the dynamic organization faces two challenges: it must have the ability to develop and maintain software to meet the customer's needs, and it must have access to software that supports the company's business processes. Both perspectives of software (external and internal) must be reliable and well maintained. Maintaining the mission-critical software of an organization is not an easy task and requires the existence of a management system for software maintenance. An adequate system for software maintenance has to satisfy a number of needs (the service criteria of a company's customers and the technical criteria of the domain), as well as maximize strategic impact and optimize the cost of software maintenance activities. This requires that the organization be committed to the continuous improvement of software maintenance processes.

Colter's observation of 1987 is still true today: "The greatest problem of software maintenance is not technical but managerial" [Col87, Ben00]. The technical issues are not far behind. There has been much research in the area of resources, processes, and tools for software maintenance. The documented problems vary according to the perspective of the author who describes the problems. There are generally two perspectives: the external perception of the customer, and the internal perception of the employees and managers who work in software maintenance. We address each in turn.

From the external perspective, a number of maintenance problems can be identified. According to Pigoski [Pig97], the cost of maintenance is too high, the speed of maintenance service is too slow, and there is

[4] CMM and CMMi are trademarks of the SEI in the United States.

difficulty in managing the priority of change requests. From the internal perspective, the work environment forces maintainers to work on poorly designed and coded software. It is also reported that there is a tremendous lack of documentation [Gla92, Huf88]. According to Bennett [Ben00], software maintainers encounter three categories of problems: perceived organization alignment problems, process problems, and technical problems.

To further exacerbate these problems, much less research has been performed for software maintenance than for development [Pig97]. There are also fewer books and research papers on the subject, and many that are commonly cited may be twenty or more years old [Lientz & Swanson 1980, Martin & McCLure 1983, Arthur 1988]. Moreover, a large number of the more recent software engineering books only refer to software maintenance marginally, as they focus on a developers' point of view [Pfl01, Pre01, Dor02, Dor02a].

Unfortunately, there is also currently a lack of specific, adaptable process improvement models for software maintenance. To address this issue and the other maintenance issues presented above, we propose a maturity model for software maintenance modeled after the CMMi© of the Software Engineering Institute [Sei02]. The contributions of this paper are threefold: First we identify the software maintenance unique activities. Second, we survey the standards, seminal literature and current maturity models for their potential contribution to maintainers. Last we introduce a proposed maturity model specific to software maintenance.

The paper is organized as follows. In section 2, we look at the state of the practice of software maintenance, and related work on software maturity models is discussed in Section 3. Section 4 presents an overview of a proposed Software Maintenance Maturity Model and its architecture. To illustrate details of this model, the goals and practices of one Key Process Area (KPA) are presented in Section 5. Finally, section 6 presents an overview of the validation process, and Section 7, conclusions and directions for future work.

2. STATE OF SOFTWARE MAINTENANCE PRACTICE

In this section, the context of software maintenance, software maintenance problems, maintenance processes and activities, and current software maintenance standards and models are discussed.

2.1 Software maintenance context

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).

Taking into account these interfaces that require 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;

We now examine each of the software maintainers interfaces (see Figure 1) in turn. The interface with the customers and users is an important one. The customer interface activities consist of negotiations and discussions about individual request priorities, service level agreements (SLAs), planning, budgeting/pricing, customer service, and user satisfaction-related activities. Users operate the software and will be involved more frequently in daily communications, which require: a) rapid operational responses to Problem Reports; b) responsiveness to inquiries about a specific business rule, screen, or report; and c) progress reports on a large number of Modification Requests.

The second maintenance interface deals with the infrastructure and operations organization communications [Iti01a, Iti01b]. Infrastructure and operations are the custodians of the infrastructure supporting the software applications. They handle all the support and maintenance issues associated with the workstations, networks and platforms and conduct activities like backups, recovery, and systems administration. The user is rarely aware of, or involved in, internal exchange of information between software maintainers and operations This interface also includes less frequent activities such as coordination of service recovery after failures or disasters in order to help restore access to services, within agreed-upon SLA terms and conditions.

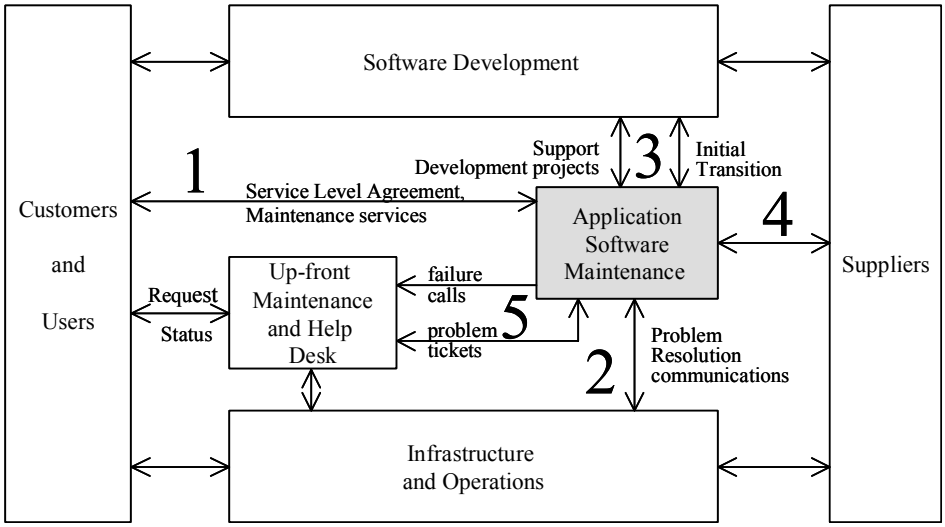

Figure 1. Software Maintainers Context Diagram.

The third interface is located between the software development and the software maintainers, and is typically initiated during the development of new software. The root cause of several maintenance problems can be traced to development, and it is recognized that the maintainers need to be involved and exercise some form of control during pre-delivery and transition [Dek92, Pig97, Ben00]. This development-maintenance interface also illustrates the contributions made by maintainers to concurrently support, and sometimes be involved in, a number of large development projects. The maintainer's knowledge of the software and data portfolios is of great value to the developers who need to replace or interface with legacy software. For example, some of the key activities would be: a) development of transition strategies to replace existing software; b) design of temporary or new interfaces; c) verification of business rules or assistance in understanding the data of existing software; and d) assistance in data migration and cutover of new software or interfaces.

The fourth interface (in Figure 1) addresses relationships with a growing number of suppliers, outsourcers, and Enterprise Resource Planning (ERP) vendors [Car94, Apr01, Mcc02]. The maintainers have a number of different relationships with suppliers, for example: a) with suppliers developing new software or configuring ERP software; b) with subcontractors who are part of the maintenance team and provide specific expertise and additional manpower during peak periods; c) with suppliers of maintenance contracts providing specific support services for their already licensed software; and d) with outsourcers who might replace, partially or completely, a function of the IT organization (*development, maintenance, or operations & infrastructure*). To ensure good service to users, software maintainers must develop some understanding of the many contract types and manage them efficiently to ensure supplier performance, which often impacts the SLA results.

The last interface (number 5 in figure 1) can be represented in many ways according to different organizational structures. Help-Desk has been found sometimes to be part of the maintenance organization, or part of the operations organization and also located in another independent product support organization. We

have chosen to represent the function independent without any specific reason. We have observed in our validation of the model that some users bypass the front-end support organizations and access directly the software maintenance personnel. This interface has been well documented as part of the CM3 Taxonomy of problem management [Kajxx] which describes in detail the problem reporting activities. To be effective, a mechanized problem resolution process is used that ensures efficient communications for quick resolution of failures. A specific user service request, sometimes called a "ticket" will typically circulate between help-desk, maintenance and operations in order to isolate a problem [*Apr01*].

2.2 Software maintenance process and unique activities

Authors report that many software organizations do not have any defined processes for their software maintenance activities [Pia01]. Van Bon [Van00] confirms the lack of process management in software maintenance and that it is a mostly neglected area. What is the source of this lack of interest in process and procedures? Schneidewind [Sch87] tells us that, traditionally, maintenance has been depicted as the final activity of the software development process. This can still be seen today in the IEEE1074-1997 standard [Iee97] which represents software maintenance as the seventh step of eight software development steps. Even today, many industrial software engineering methodologies do not even represent the software maintenance processes or activities [Sch00]. As an example, the British Telecommunications software development methodology presents maintenance as a single process at the end of software development [Btu90]. Bennet [Ben00] has a historical view of this problem, tracing it back to the beginning of the software industry when there was no difference between software development and software maintenance. Differences only became apparent during the 1970s when software maintenance life cycles started to appear. He notes that the first software maintenance life cycles consisted of three simple activities: 1) comprehension, 2) modification, and 3) validation of the software change.

The 1980s brought more extensive software maintenance process models [Ben00, Iti01a, Iti01b, Fug96]. These life cycle models represent software maintenance as a sequence of activities and not as the final stage of a software development project. The culmination of many proposals was the development of national and international standards in software maintenance during 1998 with the publication of the IEEE 1219 [Iee98] and ISO/IEC14764 [Iso98] standards that are still in use today. We are also seeing the emergence of new paradigms, such as so-called "xtreme" programming, being applied to software maintenance [Poo01]. Only time will tell whether or not support for these new approaches will be sustained [Pau02]. These two standards have been used to develop a detailed list of software maintenance activities as a first step in identifying all the key software maintenance activities.

Depending on the source of the maintenance requests, maintenance activities are handled through distinct processes. This is illustrated in Table 2 with a few examples. For each request source, a key maintenance service/process, together with registration of the related maintenance categories of work, is initiated. For example, if users are the source of the requests, then a change request related to operational use of the software and the work to be carried out can be classified within one of three maintenance services: correction, evolution (which regroups adaptive, perfective and preventive maintenance), or operational support. In some instances, a supporting process, such as service level agreement (SLA) information, will also be needed as a necessary part of the operational support activities.

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

Table 2. Examples of activities and categories of maintenance work.

2.2.1 Unique software maintenance activities

A list of unique software maintenance processes can be found in the recent version of the Software Engineering Body of Knowledge (SWEBOK) [Abr04]. It identifies a number of processes, activities, and practices that are unique to maintainers, for example:

- Transition: a controlled and coordinated sequence of activities during which a system is transferred progressively from the developer to the maintainer [Dek92, Pig97];
- SLAs and specialized (domain-specific) maintenance contracts (Apr01) negotiated by maintainers;
- Modification Request and Problem Report Help Desk: a problem-handling process used by maintainers to prioritize, document, and route the requests they receive [Ben00]; and
- Modification Request acceptance/rejection: Modification Request work over a certain size/effort/complexity may be rejected by maintainers and rerouted to a developer [Dor02, Apr01].

It also reports that a number of software engineering tools and techniques have also been adapted for the specific nature of software maintenance, including:

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

- **Software maintenance measurement:** Maintainers rely extensively on user satisfaction surveys to understand how their customers are doing [But95]. Maintainers use internal benchmarking techniques to compare different maintenance organizations and products to improve their internal processes [Abr93a, Bou96]. External benchmarking of software maintenance organizations is now becoming more popular [Abr93a, Ifp94, Her02, Isb04]. Measurement programs specific to maintainers are also being used increasingly [Gra87, Abr91, Abr93, Stp93, Sta94, Mcg95], and software estimation models specific to maintenance have been published [Abr95, Hay03, Hay04]. Pressman [Pre01] indicates that we cannot find one measure to reflect the maintainability of software, and that a number of indicators are required! This leads to some organizations using commercial tools to obtain external and internal measurements of the maintainability of software [*Boo94*, *Lag96*, *Apr00*].

- **Maintenance request repository:** An adequate information system (often shared with the operations help desk area) must be set up by the maintainer to manage the workload and track a large number of

user requests. Such a repository can become the basis for effort collection and an important component of the measurement infrastructure [Gla81, Art88, Iti01a section 4.4.7, Kaj01d, Nie02 activity 3].

- **Specific software maintainer training and education:** The following references on maintainer training and education address aspects which are specific to software maintainers [Kaj01a, Hum00, Pfl01 section 10.1 and chapter 11].

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

- **Production systems surveillance:** A maintenance organization must also put in place production systems surveillance to probe, on a daily basis, the operational environment for signs of degradation or failure. Such surveillance systems ensure that problems are identified as early as possible (ideally before the user becomes aware of them) [Iti01a section 4.4.8].

2.3 The need for updated maintenance standards

The SEEBOK initiative identified a large a number of software maintenance specific activities not covered by the current version of the international standard

2.3.1 A software maintenance process model for the maturity model

We realized early in the project the difficulty at hand in building a process model that would include the current international standard content as well as the unique activities presented by SWEBOK. We also wanted to present a process model that maintainers in the industry would quickly associate to. We propose to regroup software maintenance processes in three classes (Figure 2), the main idea is 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.).

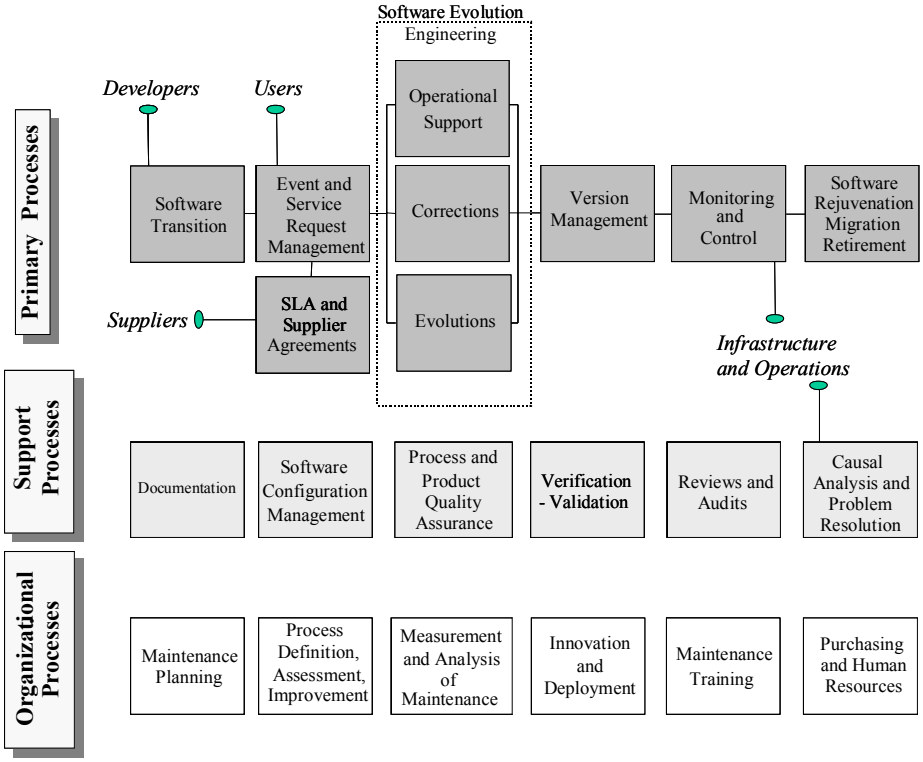

Figure 2. A classification of the Software Maintainer's Key Processes.

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. These are followed by the operational support processes and the organizational processes.

~~2.4.3~~2.3.2 Proposed software maintenance models

Based on this study, we identified that the content of the ISO/IEC 14764 and ISO/IEC 12207 standards should be updated to reflect this perspective of software maintenance. While Figure 2 would be well suited to representing the new and enhanced scope of ISO/IEC 14764 processes, we introduce Figure 3 for proposed updates to the maintenance topics of both ISO/IEC 14764 and ISO/IEC 12207. In Figure 3, we highlight the updated process view of software maintenance and show its integration into the existing process model.

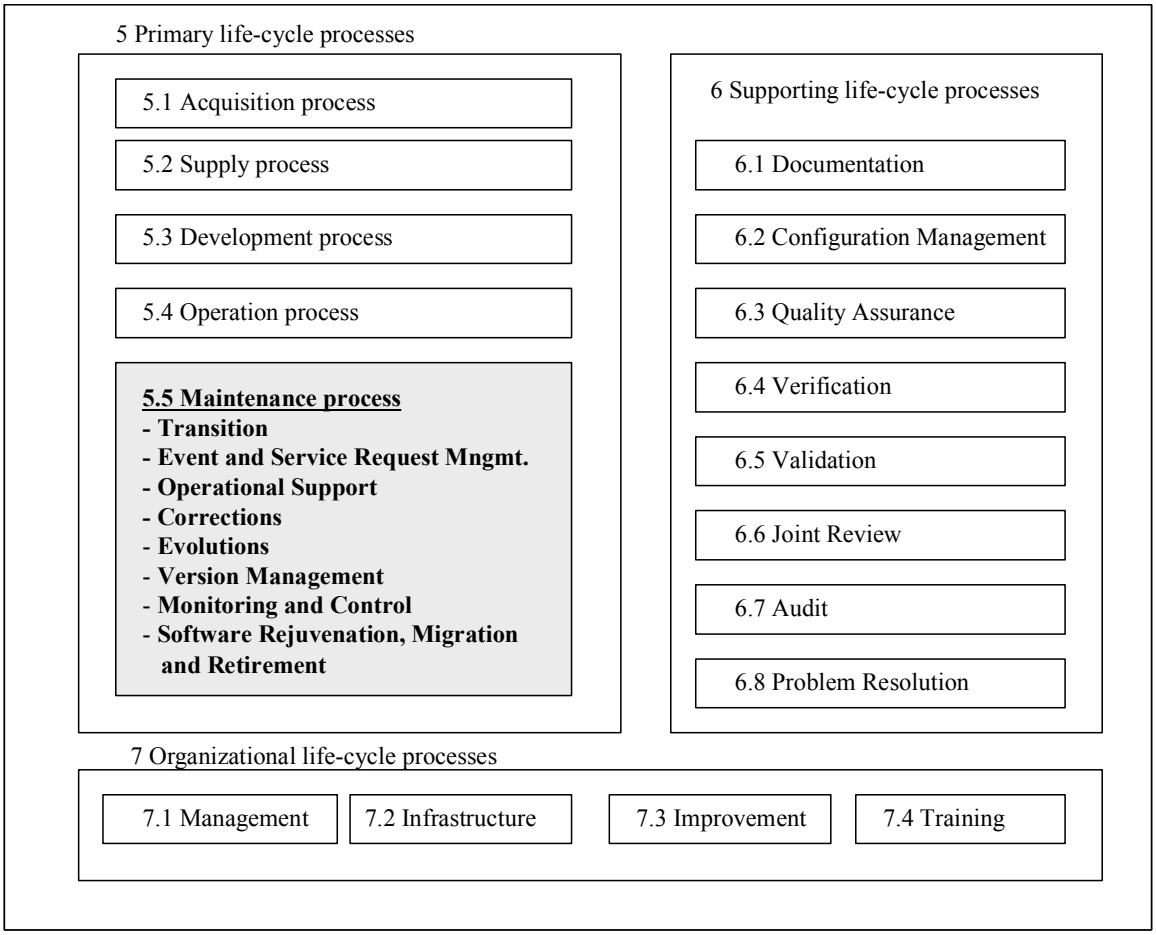

Figure 3. Proposed update to ISO/IEC 12207 maintenance processes.

3. RELATED WORK

In sections 1 and 2, citations have been provided for much of the current and past work on software maintenance models and standards. In this section, we concentrate on software engineering maturity models.

3.1 Software Maintenance Maturity Models

A literature search has not revealed any comprehensive diagnostic techniques for evaluating the quality of the maintenance process, as described by the high-level process model of Figure 2. Table 4 presents an inventory of recent proposals of software engineering process evaluation and assessment models. Each of these models has been analyzed to identify contributions that could assist maintainers. Of the thirty-four proposed models in this review, only a handful (shown in **bold** in Table 4) include documented maintenance practices, sometimes accompanied by a rationale and references. However, none of them covers the entire set of topics and concepts of the process model introduced here (in Figure 2).

Table 4. Software Engineering CMM proposals, sorted by year of publication.

Year	Software Engineering Maturity Model proposals
1991	Sei91, Tri91, Boo91
1993	Sei93
1994	**Cam94**[5], Kra94
1995	Cur95, **Zit95**
1996	Bur96 & Bur96a, Dov96, Hop96, Men96
1997	Som97
1998	Top98, Baj98, Ear98
1999	Wit99, Vet99, Sch99, Faa99, Gar99
2000	Str00, Bev00, Lud00, Luf00, **Cob00**
2001	**Kaj01d**, **Kaj01c**, Ray01, Tob01, Sri01
2002	**Sei02**, **Nie02**, Mul02, Vee02, Pom02, Raf02, Sch02, Ker02, Cra02, Win02
2003	Nas03, Doc03, Sch03a, Wid03, Rea03

Using these proposals, we completed the first inventory by adding new activities and practices to the first inventory of best practices, resulting in a much more comprehensive list of software maintenance activities covering: a) national and international standards; b) relevant software maintenance CMM proposals; and c) recognized key software maintenance references.

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

- ISO/IEC14764 [Iso98];
- IEEE1219 [Iee98];
- ISO/IEC12207 [Iso95];
- The CMMi© [Sei02];
- SWEBOK [Abr04].

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

- *Camélia* Maturity Model [Cam94];

[5] Cam94 includes and expands on Tri91 detailed practices

- Model to improve the maintenance of software [Zit95].
- CobIT [Cob00];
- Cm3-Corrective Maintenance Model [Kaj01];
- Cm3-Maintainer's Education Model [Kaj01a];
- IT Service CMM [Nie02].

We then used this list to survey 35 software maintenance specialists and managers and ask them whether or not this set of documents is complete and well suited to representing the Software Maintenance Knowledge Area. Many respondents suggested additional requirements:

- ITIL Service Support [Iti01a];
- ITIL Service Delivery [Iti01b];
- Malcolm Baldrige [Mal04];
- Iso9003:2004 [Iso04];
- Process evaluation model standard ISO/IEC 15504 [Iso02].

After reviewing the content of these additional documents, we decided that each could add value to the resulting maturity model.

3.2 CMM© and CMMi© model limitations

Our literature review (see section 2.3.1) has confirmed that some maintenance processes are unique to maintainers and not part of the software development function (see Table 3). When these unique maintenance processes are compared to the CMMi© model content, it can be observed that the CMMi© model does not explicitly address these topics. With its primary focus on project management, the CMMi© does not explicitly address the issues specific to the software maintenance function [Zit95, Apr03]. For example, in the CMMi©:

- The concept of maintenance maturity is not recognized or addressed;
- There is not sufficient inclusion of maintenance-specific practices as process improvement mechanisms;
- Maintenance-specific issues are not adequately addressed;
- Rejuvenation-related plans such as the need for redocumentation, reengineering, reverse engineering, software migration, and retirement are not satisfactorily addressed.

This was also observed in the previous version of the model, the CMM©, in 1995 by Zitouni [Zit95]. The above items are still absent from the new CMMi© version, since it maintains a developer's view of the software production process. This means that, while assessments and improvements of large maintenance activities requiring a project management structure should use the CMMi©, assessments and improvements of small maintenance activities, that do not use project management techniques, would benefit from use models better aligned to their specific characteristics, such as the **SM^{mm}** proposed next.

4. OVERVIEW OF THE MODEL AND ITS ARCHITECTURE

In this section, we introduce the software maintenance model, its scope, and its architecture, followed by some examples of the more detailed content of the model.

4.1 Constructing the model

Building maturity models is not a topic which is widely covered in the literature [Apr95, Gra98, Coa99]. The **SM^{mm}** was built by performing the steps followed during the design of the Trillium model [Tri91]. This is done by integrating practices from the key documents relevant to software maintenance best practices according to these 9 steps:

Step 1 – Practices are taken from our mappings in Appendices A & B and the high-level architecture (presented in section 4.1);

Step 2 – A mapping is performed with the CMMi© version 1.1 [Sei02]. CMMi© practices may be slightly modified to accommodate this mapping;

Step 3 – ISO 9001:2000 [Iso00] clauses, using the ISO9003:2004 [Iso04] guidelines are then reviewed one by one, added, and integrated to the model;

Step 4 - Other maturity model practices are mapped to their specific areas, depending on their coverage of the software maintenance domain (Cm3-maturity model practices of Kajko-Mattsson [Kaj01, Kaj01a], IT service CMM [Nie02], the Camélia maturity model [Cam94], and the Zitouni model [Zit95];);

Step 5 - A reference is made to IT infrastructure library best practice guidelines for Service Delivery and Service Support [Iti01a, Iti01b];

Step 6 – A lighter mapping process is performed with relevant portions of the Malcolm Baldrige examination criteria [Mal04];

Step 7 – Selected practices from CobIT [Cob00] are mapped where pertinent to software maintenance;

Step 8 - References relevant to the ISO/IEC and IEEE standards are added (ISO/IEC12207, ISO/IEC14764 and IEEE 1219);

Step 9 - Specific practices are added to provide coverage of additional areas documented in the software maintenance literature. These are based on professional benchmarks generated through the consensus of subject matter experts and validated in a peer-review process.

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

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

The same types of transformations are applied when extracting practices from other standards or best practice guides. Assignment to a given level is based on the general guidelines in section 4.6. Furthermore:

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

4.2 The resulting model

The **SM^{mm}** is presented in Table 5 (**a** and **b**). It includes 4 Process Domains, 18 KPAs, 74 Roadmaps, and 443 Practices. While some KPAs are unique to maintenance, others were derived from the CMMi© and other models, and have been modified slightly to map more closely to daily maintenance characteristics.

Table 5a. *SM*^{mm} content.

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

Table 5b. *SM*^{mm} content.

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

4.3 Purpose of the model

The SM^{mm} was designed as a customer-focused reference model, that is, a benchmark, for either:

- Auditing the software maintenance capability of a software maintenance service supplier or outsourcer; or
- Improving internal software maintenance organizations.

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

A higher capability in the **SM^{mm}** context means, for customer organizations:

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

For a maintenance organization, achieving a higher capability can result in:
a) Lower maintenance and support costs;

b) Shorter cycle time and intervals;

c) Increased ability to achieve service levels;

d) Increased ability to meet quantifiable quality objectives at all stages of the maintenance process and service.

4.4 Scope of the model

Models are often an abstract representation of reality. For a better mapping with the maintainers' reality, the **SM^{mm}** must include many of the essential perspectives of the software maintainer, and as much as possible of the maintainer's practical work context (see Figure 2). These types of models are not intended to describe specific techniques or all the technologies used by maintainers. The decisions pertaining to the selection of specific techniques or technologies are tailored to each organization. For an assessment or the design of an improvement program, users of the model must instantiate the reference model in the context of their user organization. To achieve this, professional judgment is required to evaluate how an organization benchmarks against the reference model.

4.5 Foundation of the model

The **SM^{mm}** is based on the Software Engineering Institute (SEI) Capability Maturity Model Integration for Software Engineering (CMMi©), version 1.1 [sei02] and Camélia [Cam94]. The model must be viewed as a complement to the CMMi©, especially for the processes that are common to developers and maintainers, for example: a) process definition; b) development; c) testing; d) configuration management; and e) Quality Assurance (QA) practices.

The architecture of the **SM^{mm}** (further described in section 4.6) differs slightly from that of the CMMi© version. The most significant differences are the inclusion of:

1. A roadmap category to further define the KPAs;
2. Detailed references to papers and examples on how to implement the practices.

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

1. Event and Service Request Management;
2. Maintenance Planning activities specific to maintainers (version, SLA, impact analysis);
3. SLA;
4. Software transition;
5. Operational support;
6. Problem resolution process with a Help Desk;
7. Software rejuvenation, conversion, and retirement.

4.6 SM^{mm} Architecture

The CMMi© has recently adopted the continuous representation that has been successfully used in the past by other models, such as Bootstrap [Boo91] and ISO/IEC 15504 [Iso02]. The **SM**^{mm} uses a continuous representation, as it helps to: a) conform to SPICE recommendations; b) obtain a more granular rating for each roadmap and domain; and c) identify a specific practice across maturity levels and identify its path from Level 0 (absent) to a higher level of maturity.

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

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

Figure 4. SM^{mm} Capability Levels.

The capability level definitions and the corresponding generic process attributes are described for each maturity level of the **SM**^{mm} and presented in Table 6. Section 6 presents an overview of how, over a two-year period, participating organizations contributed to the mapping of each relevant practice to a capability level in the **SM**^{mm}.

Table 6. Process characteristics by process level.

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

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

5. SAMPLE SM^{mm} DETAILS

At the detailed level for each KPA, maintenance goals and key practices have been identified based on the literature on software maintenance. This section presents, as an example, a detailed description of one of the 18 KPAs of the **SM^{mm}**: Management of Service Requests and Events. The corresponding labels for this KPA are listed in Table 7, on the basis of SPICE requirements for labeling identification.

Table 7. Example of a KPA header.

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

5.1 Overview of management of service requests and events KPA

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

These processes ensure that events, reported failures or Modification Requests and operational support requests are identified, classified, prioritized, and routed to ensure that the SLA provisions are fully met.

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

5.2 Objectives and goals

To ensure that the agreed-upon service levels are met, the objectives of this KPA are: a) to ensure that events and service requests are identified and registered daily; b) to determine the relative importance, within the current workload, of new events and service requests; and c) to ensure that the workload is focused on approved priorities. The maintainer must also communicate proactively about failures and the unavailability of software (including its planned preventive maintenance activities). This KPA covers the requirement that users be made aware of the maintenance workload, and authorize and agree on maintenance priorities. Maintainers must also oversee software and operational infrastructures as well as production software behavior (availability, performance, reliability, and stability, as well as the status of the software and its infrastructure). When priorities change, maintainers must ensure that the maintenance workload will be reassigned quickly, if necessary. The goals of this KPA are as follows:

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

Goal_2 To oversee the behavior of the software and its infrastructures during the previous 24 hours, in order to identify events that could lead to missing SLA targets;

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

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

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

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

For complete operability, this KPA requires practices from other KPAs of the **SM^{mm}**. As an example, linkages are required for: *Impact Analysis, SLA, Operational Support, and Causal Analysis & Problem Resolution.*

Once this KPA has been successfully implemented, it will be observed that:

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

5.2.1 Detailed practices

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

5.2.2 Level 0 and 1 practices

At level 0, there is only one practice:

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

Maintenance organizations operating at this maturity level perform the daily work of software maintenance without being formally accountable for their activities and priorities to the user community.

At level 1, two practices are documented in the model:

Req1.1.1 Request and event management is managed informally.

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

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

5.2.3 Level 2 practices

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

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

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

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

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

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

Maintainers classify the service requests and events according to standardized categories. Each request is assessed to determine the effort required. Pfleeger [Pfl01] adds that an impact analysis is carried out, and, in

each case, a decision is made as to how much of the standard maintenance process will be followed based on the urgency and costs of the request that can be billed to the customer.

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

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

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

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

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

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

5.2.4 Level 3 practices

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

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

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

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

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

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

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

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

Req1.3.8: Problem Reports (PRs) include detailed data related to reported failures.

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

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

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

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

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

6. VALIDATION

Activities to validate and improve the model have been taking place progressively. From 1994 to 1996, our software engineering research laboratory, sponsored by research grants from major Canadian telecommunications companies, developed an initial maturity model specific to software maintenance.

During 1998, one of the co-authors received many requests from phone company where interest in a maintenance maturity model led to the update of version 1.0 to take into account practitioners' experience, international standards, and seminal literature on software maintenance. This update activity led to version 1.5 which was published internally during 1999. Initial validation was performed in an industrial trial under the sponsorship of the Bahrain Telecommunications IS Planning Director**.** During the years 2000-01 the model was used in four process appraisals of small maintenance units (6 to 8 individuals) of this organization. Results rated three of the maintenance units as Level 1 and one as Level 2.

Following this trial, we asked 35 maintenance specialists and managers how they perceived the assessment process, the reference model, and the assessment results. The results identified software maintenance practices from the quality perspective, such as Malcolm Baldrige and ISO9000 and the European best practices guidelines [Iti01, Iti01b]. They also proved useful for the organizations, and improvement programs were initiated immediately in the areas of product measurement [Apr00] and SLAs [Apr01].

These recommendations were used to produce SM^{mm} version 2.0, the subject of two technical reports presented to the École de Technologie Supérieure during 2002. This research has been progressively made public during 2003 and 20044 in publications to software engineering conferences. Each publication was reviewed by a number of referees, who submitted comments and concurred on the industry and academic interest in this topic:

- IWSM2003 in Montréal, Canada, introduced the classification of the software maintainer's key processes;
- IASTED-SE2004 in Innsbruck, Austria, presented the model purpose, scope, and high level architecture;
- CSMR2004 in Tampere, Finland, introduced the many CMM proposals, the differences between this and the CMMi, and the model update process;
- SPICE 2004 in Lisbon, Portugal, presented the maintenance context, the model's generic attributes, the detailed KPAs, as well as an example of one of the model's detailed practices for the Management of Service Requests and Events KPA.

7. CONCLUSIONS AND FUTURE WORK

This paper has presented a software maintenance model (**SM^{mm}**) developed to assess and improve the quality of the software maintenance function. The **SM^{mm}** architecture is based on the model developed by the SEI of the Carnegie Mellon University of Pittsburgh to evaluate and improve the process of software development. The identification of key differences between the development and the maintenance function was based on industry experience, international standards, and the literature on software maintenance. To illustrate the detailed content of the **SM^{mm}**, we have presented the goals of a KPA, together with the detailed practices from Levels 0 to 3.

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

As a result of the very helpful Spice 2004 conference in Lisbon, a number of organizations in France, Brazil, Switzerland, and Canada have contacted us to undertake cooperative activities concerning

the use of the model in their organizations. Additional field study is required to further validate this maintenance model. This will ensure that the key practices suggested by maintenance experts or described in the literature are positioned at the correct level of maturity within the model. Other future work will include application across various domains to ensure general applicability.

ACKNOWLEDGMENTS

Our thanks are extended to Mr. Dhiya Al-Shurougi for his leadership in conducting the industrial trial of this model, and our thanks also to the following individuals who contributed to the model content: Ernst & Young – South Africa (Nic Grobbelar), Orange Switzerland (Luc Girard), Celepar – Brazil (Vanrerlei Vilhanova Ortêncio), Delmarva Power (Alex E.Dobkowsi), SPRINT (John Gartin), Jane Huffman Hayes (University of Kentucky), André Lariviere, Luc Chaput, Mohammed Zitouni, Dr. Jacques Bouman, David Déry, Nazeem Abdullatief, Paul Goodwin, Roxanne White, Peter Keeys, Normand Lachance, Sujay Deb, Garth George, Ravi Kalyanasundaram, Ganesh Ayyer, Phil Trice, Andy Crowhurst, Mario Ferreira, Ian Robinson, Taha Hussain, Dr. Talib Salman, and Ali Al-Jalahma.

REFERENCES

[1] [Abr04] Abran A, Moore JW (Exec. Eds), Bourque P, Dupuis R (Eds). *Guide to the Software Engineering Body of Knowledge (SWEBOK) – 2004 Version*. IEEE Computer Society: Los Alamos, California, 2004; to be published.

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

[3] [Abr93] Abran A, Nguyenkim, H. Measurement of the maintenance process from a demand-based perspective. *Journal of Software Maintenance: Research and Practice* 1993;**5**(2):63-90.

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

[5] [Abr95] A. Abran, M. Maya, A Sizing Measure for Adaptive Maintenance Work Products, ICSM-95, Opio, France, Oct. 1995, pp. 286-294.

[6] [Apr04] A.April, A.Abran, R.Dumke, "SM^{cmm} to Evaluate and Improve the Quality of Software Maintenance Process: Overview of the model," SPICE 2004 Conference on Process Assessment and Improvement, Critical Software SA, The Spice User Group, Lisbon (Portugal), Apr. 27-99, 2004, pp. 19:32.

[7] [Apr04a] A.April, A.Abran, R.Dumke, "Assessment of Software Maintenance Capability: A model and its Architecture," CSMR 2004, 8th European Conference on Software Maintenance and Reengineering, Tampere (Finland), Mar. 24-26, 2004, pp. 243-248.

[8] [Apr03] April A, Abran A, Bourque P. Analysis of the knowledge content and classification in the SWEBOK chapter: Software maintenance. Technical Report 03-001 of the ETS Software Engineering Laboratory, 2003, 12 pp.

[9] [Apr01] April A, Bouman J, Abran A, Al-Shurougi D. (2001). Software maintenance in a SLA: Controlling the customer expectations. Proceedings of the Fourth European Software Measurement Conference, FESMA May 8-11 2001. Publisher Technologish Instituut vzw, Heidleberg, Germany, pp. 39-47.

[10] [Apr00] April, A., Al-Shurougi, D. (2000). Software Product Measurement for supplier Evaluation, FESMA-AEMES Software Measurement Conference, Madrid (Spain), October 18-20, 2000, http://www.lrgl.uqam.ca/publications/pdf/583.pdf (link tested May 11, 2004).

[11] [Apr95] April, A., Coallier, F. TRILLIUM V3.0: A model for the Assessment of Telecom Software Development Capability, Proceedings 2nd International SPICE Symposium, edited by T.P. Rout, AQRI, June 1-2, 1995, Brisbane (Australia), pp. 79-88.

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

[13] [Baj98] Bajers F. How to introduce maturity in software change management. Technical Report R98-5012, Department of Computer Science, Aalborg University: Denmark, 1998; 9 pp. http://www.cs.auc.dk/~gobe/Publications/Bendix98b/IR.html (link tested on May 11, 2004).

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

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

[16] [Bev00] Bevans, N. Introduction to Usability Maturity Assessment, Serco Usability Services, UK, 2000, 10 pp. http://www.usability.serco.com/trump/documents/Maturity_assessment.ppt.pdf (link tested on May 11, 2004)

[17] [Boo91] European Commission. Bootstrap. Esprit project #5441, European Commission: Brussels, Belgium, 1991; 26 pp.

[18] [Boo94] Booch, G. Bryan, D. Software Engineering with Ada, Third Edition, Benjamin/Cummings Publishing Company, Redwood City, California, 1994, 560 pp.

[19] [Bou96] Bourque, P. Maintenance Benchmarking Projects: An Overview. International Software Benchmarking: Engineering and Measurement Issues. 1996. Montreal: Université du Québec Montréal, 21pp.

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

[21] [Bur96] Burnstein I, Suwannasart T, Carlson C. Software test process evaluation and improvement. Test Conference Proceedings International, October 20-25, 1996, pp. 581-589.

[22] [Bur96a] Burnstein I, Suwannasart T, Carlson C. Developing a testing maturity model: Part II. In Crosstalk Journal. U.S. Department of Defense; UT, September 1996; 9 pp. http://www.improveqs.nl/pdf/Crosstalk%20TMM%20part%202.pdf [link tested on April 23, 2004].

[23] [But95] Butler, K. (1995). The Economic Benefits of Software Process Improvement, Crosstalk Journal, U.S. Department of Defense; UT, 8(7), July issue, 14-17 pp..

[24] [Cam94] Camélia. Modèle d'évolution des processus de développement, maintenance et d'exploitation de produits informatiques, Version 0.5. Projet France-Québec: Montréal, Canada, 1994; 115 pp.

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

[26] [Car94] Carey D. Executive round-table on business issues in outsourcing – Making the decision. CIO Canada. IT World Canada Inc.; Scarborough (Ontario), June/July 1994; 21 pp.

[27] [Coa99] Coallier, F., Mayrand, F., Lague, B. (2000). Risk Management In Software Product Procurement, in Elements of Software Process Assessment and Improvement, IEEE CS Press, 23-44.

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

[29] [Col87] Colter, M., "The Business of Software Maintenance," *Proc. 1st Workshop Software Maintenance,* University of Durham, Durham (England), 1987.

[30] [Cra02] Crawford, J.K. Project management maturity model, Providing a proven path to project management excellence. Marcel Dekker/Center for business practices: ISBN:0824707540, January 2001, 121 pp.

[31] [Cur95] Curtis, B., Hefley, W., Miller, S., (1995) People Capability Maturity Model, Software Engineering Institute, CMU/SEI-95-MM-02, 445 pp.

[32] [Dek92] Dekleva SM. Delphi study of software maintenance problems. *Proceedings of 1992 IEEE Conference on Software Maintenance, ICSM 1992.* IEEE Computer Society: Orlando, Florida, 1992:10-17.

[33] [Doc03] US Department of Commerce. IT Architecture Capability Maturity Model (ACMM), 2003. http://www.opengroup.org/architecture/togaf8-doc/arch/p4/maturity/mat.htm (link tested on April 9, 2004).

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

[35] [Dor02a] M. Dorfman and R. H. Thayer. *Software Engineering, 2nd Edition. Volume 2 – The Supporting Processes*. Edited by Richard H. Thayer and Mark Christensen, IEEE Computer Society Press, 2002, ISBN 0769515576.

[36] [Dov96] Dove R, Hartman S, Benson S. A change proficiency maturity model, An agile enterprise reference Model with a case study of Remmele Engineering. *Agility Forum, AR96-04*. Publisher: Agility Forum, New Mexico, 1996; 109 pp.

[37] [Ear98] Earthy, J. Usability Maturity Model: Processes, Process Contracting Limited, version 1.2, 27/12/98, http://www.ipo.tue.nl/homepages/mrauterb/lecturenotes/USability-Maturity-Model%5B1%5D.PDF (link tested 9 April, 2004).

[38] [Faa99] Federal Aviation Administration. The Federal Aviation Administration integrated Capability Maturity Models (FAA-CMMi®), 10/7/99. http://www.faa.gov/cm/RegionalMinutes/FAA%20iCMM/ (link tested April 9, 2004).

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

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

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

[42] [Gar99] Garcia Diez, A.B., Lerchundi, R. TeleSpice Process Model, ESI White Paper, European Software Institute (ESI-1999-telespice-ext), December 1999, 22 pp.

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

[44] [Gla81] Glass, R.L, Noiseux, R.A. Software Maintenance Guidebook, prentice-Hall. January 1981, ISBN: 0138217289.

[45] [Gra98] Gaydon, A.W., Nevalainen, R., et al. (1998). The Reference Model, in SPICE: An Empiricist's Perspective, Proceedings of the Second IEEE International Software Engineering Standards Symposium, pp. 75-97.

[46] [Gra87] Grady, R. Caswell, D. Software Metrics: Establising a Company-wide Program. Englewood Cliffs, NJ, Prentice-Hall, 1987.

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

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

[49] [Hay03] Hayes J Huffman, Mohamed N, Gao T. The observe-mine-adopt model: an agile way to enhance software maintainability. *Journal of Software Maintenance and Evolution: Research and Practice* 2003:**15**(5): 297 – 323.

[50] [Hay04] Hayes J Huffman, Patel S, Zhao L. A metrics-based software maintenance effort model. *Proceedings of the 8th European Conference on Software Maintenance and Reengineering, CSMR 2004*. IEEE Computer Society: Tampere, Finland, 2004, pp. 254-258.

[51] [Her02] Hernu, M. Solomon Associates, Inc. *Using Benchmarking Data Effectively*, http://www.mt-online.com/current/03-01mm01.html (link tested on 11th May 2004).

[52] [Hop96] Hopkinson, J.P. System Security Engineering Maturity Model, EWA-Canada. http://philby.ucsd.edu/~cse291_IDVA/papers/hopkinson.system_security_engineering_capability_maturity_model.pdf (link tested on 9 April 2004).

[53] [Huf88] Huffman J, Burgess C. Partially Automated In-line Documentation (PAID): Design and implementation of a software maintenance tool. *Proceedings of the 1988 IEEE Conference on Software Maintenance, ICSM 1988*. IEEE Computer Society: Phoenix, Arizona, 1988:60-65.

[54] [Hum00] Humphrey, W. Managing Technical People – Innovation, teamwork and the software Process, SEI Series in Software Engineering, Addison-Wesley, 2000: 496 pp.

[55] [Iee97] Institute of Electrical and Electronics Engineers. *IEEE Standard for Developing Software Life Cycle (Software), Standard 1074*. Institute of Electrical and Electronics Engineers: New York NY, 1997;

[56] [Iee98] Institute of Electrical and Electronics Engineers. *IEEE Standard for Software Maintenance, Standard 1219*. Institute of Electrical and Electronics Engineers: New York NY, 1998; 47 pp.

[57] [Ifp94] International Function Point User Group, Guidelines to Software Measurement, version 1.0, Westerville (Ohio), 1994, 117 pp.

[58] [Isb04] International Benchmarking Software group (2003). Maintenance and Support Collection Package, Version 1.0, http://www.isbsg.org.au/html/index2.html (link tested on May 11, 2004).

[59] [Iso00] ISO9001:2000, Quality Management Systems – Requirements, International Organization for Standardization, Third edition 2000-12-15, International Organization for Standardization, Geneva, Switzerland.

[60] [Iso95] International Standards Organization. Information Technology – Software Life Cycle Processes, ISO/IEC Standard 12207. International Organization for Standardization: Geneva, Switzerland, 1995; 57 pp.

[61] [Iso98] International Standards Organization. Software Engineering-Software Maintenance, ISO/IEC Standard FIS 14764. International Organization for Standardization: Geneva, Switzerland, 1998; 38 pp.

[62] [Iso02] International Standards Organization. Information Technology – Process Assessment – Part 1 to 5, ISO/IEC Standard 15504. International Organization for Standardization: Geneva, Switzerland, 2002-2004.

[63] [Iso04] International Standards Organization. Software and System Engineering – Guidelines for the Application of ISO9001:2000 to Software, ISO9003:2004, Geneva, Switzerland, 2004; 42 pp.

[64] [Iti01a] Central Computer and telecommunications Agency. Service Support. In Information Technology Infrastructure Library. HSMO Books: London, UK, 2001; 269 pp.

[65] [Iti01b] Central Computer and telecommunications Agency. Service Delivery. In Information Technology Infrastructure Library. HSMO Books: London, UK, 2001; 378 pp.

[66] [Jon96] Jones, C. Applied Software Measurement. New York, NY:McGraw-Hill, 1996, 618pp.

[67] [Kaj01] Kajko-Mattsson M. Corrective maintenance maturity model: Problem Management, Report Series 01-015, ISSN 1101-8526, Stockholm University: Stockholm, Sweden, 2001; 290 pp.

[68] [Kaj01a] Kajko-Mattsson, M., Forssander, S., Olsson, U., Corrective Maintenance Maturity Model: Maintainer's Education and Training, in Proceedings, International Conference on Software Engineering, IEEE Computer Society Press: Los Alamitos, CA, 2001; pp. 610-619.

[69] [Ker02] Kerzner H. Strategic planning for project management using a project management maturity model. John Wiley & Sons: New York, NY, 2002; 255 pp.

[70] [Kra94] Krause MH. Software - A maturity model for automated software testing. In Medical Devices and Diagnostic Industry Magazine. Canon Communications llc; December 1994; http://www.devicelink.com/mddi/archive/94/12/014.html, 8 pp. (link tested 11 May 2004)

[71] [Lag96] Lague, B., April, A. Mapping of the ISO9126 Maintainability Internal Metrics to an industrial research tool, SESS, Montreal, 1996, October 21-25, http://www.lrgl.uqam.ca/sponsored/ses96/paper/lague.html (link tested 11 May 2004)

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

[73] [Lud00] Ludescher G, Usrey MW. *A New Renaissance: Multinational, Multidisciplinary & Performance-Focused,* Proceedings of the First International Research Conference on Organizational Excellence in the Third Millennium, (R. Edgeman, editor), pp. 168-173, Estes Park, CO, August 2000.

[74] [Luf00] Luftman J. Assessing business-IT alignment maturity. *Communications of the Association for the Information Systems*, December 2000; **4**(14), http://www.dmst.aueb.gr/gr/Courses/5sem/23_dioik_epix_texn/PPTS/paper2B5.pdf (link tested 11 May 2004).

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

[76] [Mag97] Magee, S., Tripp, L. Guide to Software Engineering Standards and Specifications, Artech House, Boston-London. December, 1997 ISBN: 1580532519.

[77] [Mcc02] McCracken B. Taking control of IT performance. InfoServer LLC: Dallas, Texas, October 2002; 6 pp. http://www.outsourcing-information-technology.com/control.html (tested on 11 May 2004).

[78] [Mcg95] McGarry, J. Practical Software Measurement: A Guide to Objective Program Insight, Department of Defense, September 1995.

[79] [Men96] Menk, C.G. System Security Engineering Capability Maturity Model (SSE-CMM), Department of Defense. http://csrc.nist.gov/nissc/1996/papers /NISSC96/paper010/CMMTPEP.PDF (link tested 9 April 2004).

[80] [Moo98] Moore, J. *Software Engineering Standards: A User's Road Map*. IEEE Computer Science Press, Los Alamitos, 1998, ISBN: 0-8186-8008-3, 297 pp.

[81] [Mul02] Mullins C. The capability model – from a data perspective. The Data Administration Newsletter, 2002. www.tdan.com/i003fe04.htm, (link tested on October 30, 2003).

[82] [Nas03] NASCIO. NASCIO Enterprise Maturity Model, version 1.3, December 2003. https://www.nascio.org/hotIssues/EA/EAMM.pdf (link tested on April 9, 2004).

[83] [Nie02] Niessink F, Clerk V, van Vliet H. *The IT service capability maturity model, release L2+3-0.3 draft.* Software Engineering Research Centre: Utrecht, The Netherlands, 2002; 133 pp. http://www.itservicecmm.org/doc/itscmm-l23-0.3.pdf [Link tested on July 2nd 2004].

[84] [Pau02] Paulk, M. (2002). Agile Methodologies and Process Discipline, Crosstalk Journal, U.S. Department of Defense; UT October 2002 http://www.stsc.hill.af.mil/crosstalk/2002/10/paulk.html (link tested July 2nd 2004).

[85] [Pfl01] Pfleeger, S.L. *Software Engineering—Theory and Practice.* Prentice Hall, 2nd ed., 2001, 659 pp.

[86] [Pia01] Piattini, M., Villalba, J. et al. Mantenimiento del software, editor: Alfaomega-Rama. Mexico. ISBN: 970-15-0730-4, 336 pp.

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

[88] [Pom02] Projxsoft. Project Organization Maturity Model (POM2). http://www.projxsoft.com/default.asp?nc=2053&id=4 (link tested on 3rd May 2004).

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

[90] [Pre01] Pressman RS. *Software Engineering A Practitioner's Approach.* McGraw-Hill, New York, NY, 2001: 860 pp.

[91] [Raf02] Raffoul W. The outsourcing maturity model. Meta Group, 2002. http://techupdate.zdnet.com/techupdate/stories/main/0%2C14179%2C2851971-2%2C00.html#level1 (link tested on 3rd May 2004).

[92] [Ray01] Rayner P, Reiss G. The programme management maturity model. The Programme Management Group, 2001. http://www.e-programme.com/events/pmmm.htm [link tested on 23 April 2004].

[93] [Rea03] Ream, S.W. (2003). The business continuity Maturity Model, Virtual Corporation. http://www.virtual-corp.net/ (link tested on 3rd May 2004).

[94] [Sch03] Schach S, Jin B, Wright D, Heller G, Offutt AJ. Determining the distribution of maintenance categories: survey versus empirical study. *Kluwer's Empirical Software Engineering* 2003; **8**(4):351-365.

[95] [Sch03a] Schekkerman, J. Extended Enterprise Architecture Maturity Model, Institute for Enterprise Architecture Development. 2003. http://www.enterprise-architecture.info/Images/E2AF/Extended%20Enterprise%20Architecture%20Maturity%20Model%20_E2AMM_09-2003.pdf (link tested on 3 May 2004).

[96] [Sch02] Schlichter J. An introduction to the emerging PMI organizational project management maturity model, 2002. http://www.pmi.org/prod/groups/public/documents/info/pp_opm3.asp (link tested on 3rd May 2004).

[97] [Sch99] Schmietendorf A, Scholz A. The performance engineering maturity model at a glance. Metrics News 1999; 4(2): 342pp.

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

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

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

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

[102] [Sei91] Paulk, M.C., Curtis, B., et al. Capability Maturity Model for Software, SEI, CMU/SEI-91-TR-24, ADA240603, August 1991: 61 pp.

[103] [Som97] Sommerville I, Sawyer P. Requirements engineering: A good practice guide. John Wiley & Sons, Chichester, 1997: 391 pp.

[104] [Sri01] Sribar V, Vogel D. The capability maturity model for operations. Metagroup. http://www.metagroup.com/metaview/mv0452/mv0452.html [link tested on April 23, 2004].

[105] [Sta94] Stark, G.E. Kern, L.C. and Vowell, C.V. A Software Metric Set for Programme Maintenance Management. Journal of Systems and Software, March 1994, pp. 239-249.

[106] [Stp93] St-Pierre, D. Integration of Maintenance Metrics. in IEEE International Conference on Software Maintenance - ICSM. 1993. Montréal, Québec: pp. 374-375.

[107] [Str00] Stratton RW. The earned value management maturity model. http://www.mgmt-technologies.com/evmtech.html [link tested on April 23, 2004].

[108] [Tob01] Tobia E, Glynn M. E-business, can we control it? , e-business maturity model. 14th Utility Coal Conference, PricewaterhouseCoopers. www.utilitycoalconference.com [link tested on April 23rd 2004].

[109] [Top98] Topaloglu, N.Y. Assessment of Reuse Maturity, Ege University Computer Engineering Department, Bornova-Izmir, Turkey, 1998, http://people.cs.vt.edu/~edwards/icsr5/bpt/yasemin.htm (link tested 11 May 2004).

[110] [Tri91] Trillium, Model for the Telecom Product Development & Support Process Capability, Bell Canada, version 1.0, 91/08,Version 3.0 available at http://www2.umassd.edu/swpi/BellCanada/trillium-html/trillium.html (link tested 11 May 2004)

[111] [Van00] Van Bon, J. World Class IT Service Management Guide 2000, Van Bon (eds.) ten Hagen & Stam Publishers, The Netherlands ISBN 90-76383-46-4, 2000

[112] [Vee02] Veenendaal V, Swinkels R. Guideline for testing maturity: Part 1: The TMM model. Professional Tester, Vol. 3, Issue 1, 2002. http://www.improveqs.nl/tmmart.htm (link tested on 3rd May 2004).

[113] [Vet99] Vetter R. The network maturity model for Internet development. *IEEE Computer* 1999; **132**(10):117-118.

[114] [Wid03] Widdows, C., Duijnhouwer, F-W. Open Source Maturity Model. Cap Gemini Ernst & Young,.http://www.seriouslyopen.org/nuke/html/modules/Downloads/osmm/GB_Expert_Letter_Open_Source_Maturity_Model_1.5.1.pdf , 2003 18 pp. (link tested on 3rd May 2004).

[115] [Win02] Windley, P.J. (2002). eGovernment Maturity, CIO State of Utah. http://www.windley.com/docs/eGovernment%20Maturity.pdf (link tested on 3rd May 2004).

[116] [Wit99] Wichita State University. Enterprise Engineering Presentation, Capability Model of Business Process Reengineering. Course IE80I. Wichita, 1999; 31 pp.

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

AUTHORS' BIOGRAPHIES

Alain April is professor of Software Engineering at the École de Technologie Supérieure – ÉTS, Université du Québec, Montréal, Canada. He is pursuing a doctorate at the Otto von Guericke University of Magdeburg, Germany. His research interests are: software maintenance, software quality, and the database. He has worked in the IT industry for more than 20 years in the telecommunications industry. Professor April has contributed to the internal measurement of software of ISO9126 (part 3) and is the associate editor of the SWEBOK (Software Engineering Body of Knowledge) software maintenance and software quality chapters. E-mail: alain.april@etsmtl.ca

Jane Huffman Hayes is assistant professor of Software Engineering in the Computer Science department of the University of Kentucky. Previously, she was a Corporate Vice President and Operations Manager for Science Applications International Corporation. She received an MS degree in computer science from the University of Southern Mississippi and a PhD in Information Technology from George Mason University. Her research interests include software verification and validation, requirements engineering, software testing, and software maintenance. E-mail: hayes@cs.uky.edu

Alain Abran is professor of Software Engineering at the École de Technologie Supérieure – ÉTS, Université du Québec. As the director of the Software Engineering Research Laboratory, he manages many research projects for industry and international professional associations www.lrgl.uqam.ca .

Much of Dr. Abran's work has been used in recent ISO standards. He was the international secretary of the Software Engineering sub-committee from 2001 to 2003, and is currently the executive co-editor of the SWEBOK (Guide to the Software Engineering Body of Knowledge).

Prior to joining the university in 1994, Dr. Abran worked for more than twenty years in system design, project management, and in numerous IT management positions in the Canadian financial industry. Dr. Abran received his PhD from the École Polytechnique de Montréal and has MSc degrees from the University of Ottawa in electrical engineering (computer sciences) and in administration (MSc management). E-mail: alain.abran@etsmtl.ca.

Reiner Dumke holds a PhD on the operational efficiency of large-scale database systems. Since 1994, he has been a full professor in software engineering at the University of Magdeburg. He his the leader of the German Interest Group on software metrics within the German Informatics Society (GI) and founder of the Software Measurement Laboratory at the University of Magdeburg. He is co-editor of the Metrics News Journal and the publisher and editor of more than 30 books about programming techniques, software metrics, metric tools, software engineering foundations, component-based software development and web engineering. E-mail: dumke@ivs.cs.uni-magdeburg.de