Motivating the Corrective Maintenance Maturity Model (CM$^3$)

Mira Kajko-Mattsson
Software Maintenance Laboratory
Department of Computer and Systems Sciences
Stockholm University/Royal Institute of Technology
Electrum 230
SE-164 40 KISTA
mira@dsv.su.se

Abstract
Many process models have been hitherto suggested. Very few of them, however, deal with the most utilised process today – the maintenance process. In this paper, we present and motivate the “Corrective Maintenance Maturity Model” (CM$^3$) - a process model entirely dedicated to only software maintenance.

Keywords
Maintenance, maturity, process model, maintenance definition, corrective maintenance.

1 Introduction
Maintenance is our “Software Cinderella” – a neglected stepsister to Software Development. We do not appreciate it, but we cannot live without it. The real Cinderella, however, dropped her shoe only once, and soon after became a queen. Our “Software Cinderella”, on the other hand, has dropped her shoe(s) many times. Her voyage towards the “royal” recognition has not gained much success yet.

In recent years, however, the software community has started to recognise software maintenance as a crucial discipline within software engineering. This is due to the fact that we do maintain more than develop today. Maintenance has become the dominating cost factor in most of the software organisations. Despite this, the majority of the present process models are still dedicated to software development.

In this paper, we present and motivate the “Corrective Maintenance Maturity Model” (CM$^3$) – a process model entirely committed to only software maintenance.

1.1 Background
Maintenance is just as old as development. As soon as we started developing software, we had to maintain it. The importance of maintenance, however, has for many years been ignored and/or belittled. This has lead to the fact that maintenance is still considered to be a very immature area. It has become one of the most complex, costly, difficult, and the least understood activities within software engineering. Maintenance still suffers from many uncertainties and problems. Some of them are (1) choice of an appropriate name, (2) definition of maintenance, (3) maintenance categorisation, (4) disagreement in the use of terminology, and (5) shortage of maintenance process models.

There has been a sustained debate for about two decades on the appropriateness of the term “maintenance” [18]. According to [4, 8, 9, 25, 27, 33], maintenance is not the right name for this software engineering discipline. Other terms have been suggested such as software support, software evolution, co-evolution, continuation engineering, production monitoring, systems control, post implementation development, system tuning, program evolution, application software support [4, 8, 9, 25, 30].

There is a disagreement on the definition of software maintenance. According to [29, 30], the definition suggested by the IEEE [14, 15] gives a distorted view of maintenance. It states that maintenance is entirely a post-delivery activity. There is also a controversy on the choice of maintenance scope, its constituents, time span, and on drawing a dividing line between software development and software maintenance [28, 29, 31]. No wonder that there have been so widely varying costs of software maintenance spanning between 40%-90% of the total software life-cycle cost [1, 2, 3, 12, 20, 26].

The categorisation of maintenance as suggested by the IEEE standard is not even mutually exclusive [5, 19]. Close scrutiny reveals inconsistency in the way the software community interprets and applies the IEEE definition of preventive and perfective maintenance [14, 15]. Analysis contrasting current practice with the IEEE definition of maintenance and its categories [14, 15] reveals overlap between preventive and perfective maintenance [5, 19]. To remedy this, new categorisations have been recently suggested in [6, 24].

The non-uniformity in the use of maintenance concepts has been notified [6, 17]. Terminology used by industry and academia differ greatly. Even within one and the same organisation, one uses different terms for one and the same concept. Attempts have been made to
conceptualise and clarify the maintenance terms in [16, 17, 24].

There is a shortage of maintenance process models. To the knowledge of the author of this paper, there are only two internationally recognised maintenance process models – IEEE 1219 [15], and the recently suggested ISO/IEC 14764 standard [11]. The other models such as the ISO/IEC 12207 [10], CMM [7] and TickIT [32] contain maintenance as one of its many constituents barely touching on the subject of software maintenance.

The present maintenance process models are too general. By covering all maintenance categories such as corrective, adaptive, perfective and preventive, they do not help in acquiring a deep understanding of the scope of each maintenance category. Managing such a huge and complex domain by one and the same process model has lead to the aforementioned uncertainties and many unanswered questions.

One way to remedy this is to construct process models entirely dedicated to each maintenance category. Maintenance categories differ too much in order to be lumped together under one and the same process model. We believe that each maintenance category deserves its own process model (a specialised model). This would offer a good opportunity to scrutinise the definition of each maintenance category, its name, goal, scope, and most importantly, this would lead us towards an objective understanding of software maintenance. Specialised models would also aid in agreeing on a single classification system for each maintenance category and its inherent activities. To the knowledge of the author of this paper, there are no process models explicitly defined for each maintenance category.

1.2 Contribution

In this paper, we present and motivate a process model for corrective maintenance. We call it Corrective Maintenance Maturity Model (CM³). CM³ is presently being developed at ABB by Software Maintenance Laboratory – a collaboration between ABB and Department of Computer and Systems Sciences at Stockholm University and Royal Institute of Technology in Sweden. CM³ is the result of our empirical study of industrial corrective maintenance processes utilised at ABB Automation Products AB (ABB APR) and ABB Robotics AB (ABB ROP).

Our primary goal is to suggest a fine-grained process model providing a maximal visibility into the corrective maintenance process. Our other goals are (1) to establish a common forum for communicating about corrective maintenance, (2) to enable industrial organisations examine their own practices and compare notes, (3) to provide guidance to industrial organisations in the process of building or improving their corrective maintenance processes, and (4) to provide a pedagogical tool for universities and industrial organisations in the process of educating their students and software engineers within the area of corrective maintenance.

CM³ is entirely dedicated to corrective maintenance. In the future, we hope that CM³ will be part of an even greater model covering all maintenance categories. Preliminarily, we call it Maintenance Maturity Model (M³).

2  CM³ - Definition Maturity Model

Definition of maintenance and corrective maintenance is fundamental for building process models. Due to the fact that confusion prevails concerning the definition of maintenance, we have created our own one by compiling the international protests [28, 30], and the definitions suggested by Pigoski’s [29] and the IEEE [14, 15]. Our definition of maintenance and of corrective maintenance reads as follows.

Software maintenance is the totality of activities required to provide cost-effective support to a delivered software system. These activities are performed during the (1) predelivery/prerelease, (2) transition/delivery, and (3) postdelivery/postrelease stages. Predelivery/prerelease activities include planning for the transition/delivery and postdelivery/postrelease operation. Transition/delivery stage encompasses the activities required for a smooth transfer of a software system from predelivery/prerelease stage to postdelivery/postrelease one. Postdelivery/postrelease activities include software modification according to the following maintenance categories:

Corrective Maintenance: Activities performed to attend to defects in software systems.

Other Maintenance: Activities required to conduct other maintenance categories.

Our definition of corrective maintenance differs from the IEEE one with respect to its verbalisation. According to the IEEE standard [14], “maintenance is performed to correct faults”. A fault is a defect found in an executable code [13, 14], and a defect is an anomaly found in a software product [13]. Under the IEEE definitions of defect and fault, we understand a fault to be a sub-entity of a defect. We feel that the IEEE has chosen an inappropriate wording when defining corrective maintenance. Within corrective maintenance, we do not only correct faults in an executable code, but also design defects, documentation defects, test case specification defects and so forth. To remedy this, we have reworded the IEEE definition of corrective maintenance to be explicit enough that we mean all kinds of defects found in a software system.

3  CM³ - Life-Cycle Perspective

According to [29, 30], the IEEE definition of maintenance has done disservice to the software community by claiming that maintenance is entirely a postdelivery activity. Attempts to remedy this have been made by Pigoski [29]. Pigoski has suggested two maintenance phases such as predelivery and postdelivery.
He has also suggested a transition phase to be taken between those two.

We have designated three maintenance phases (a) predelivery/prerelease phase, (b) transition/delivery phase, and (3) postdelivery/postrelease phase (see Figure 1).

3.1 Predelivery/Prerelease phase

According to [12, 29, 30, 31], software transition/delivery and postdelivery/postrelease software maintenance are greatly influenced by the level of maintainers’ involvement during the predelivery/prerelease phase. Irrespective of who is going to maintain the software system (either the development organisation or a separate maintenance organisation), there is a need for a predelivery maintenance phase. Certain maintenance activities must be namely conducted then. These activities are for the most part of a preparatory nature for the later phases. During this phase, we must create and realise a maintainability plan and a maintenance plan, we must prepare for the transition/delivery phase, and other things.

3.2 Transition/Delivery phase

Software transition/delivery is a controlled and co-ordinated sequence of activities during which (1) a newly developed software system is transferred from the organisation that conducted development to both the customer and maintenance organisations, or (2) a modified software system is transferred from the organisation that conducted modification to both the customer and maintenance organisations.

3.3 Postdelivery/Postrelease phase

Postdelivery phase begins after the software product has been delivered to the customer. It includes the sum of the activities to conduct changes to software systems. The changes are implemented incrementally in subsequent product releases (see Figure 1). Hence, the postdelivery phase consists of a series of subsequent postrelease phases.

Figure 1. CM³: Roadmap. Life-cycle perspective.

Figure 2. CM³ Roadmap: product perspective.

Usually, the creation of one major software release corresponds to the modification of software due to modification requests for either enhancements and/or adaptations. When implementing these modifications, however, we should not forget to treat this sub-phase as some kind of a predelivery phase for the next release. In this phase, we should build in and preserve the maintainability of the product. To distinguish this phase from the development phase, we call it prerelease phase.

4 CM³ - Product Perspective

When viewing corrective maintenance from the product perspective, CM³ identifies two phases: direct and indirect corrective maintenance (see Figure 2). During direct corrective maintenance, all reported software problems are successively attended to as they arrive to the corrective maintenance process. The correction of one set of problems leads to the creation of a new revision of a software product. This new revision is sent to the customers affected by the resolved software problems. It is also announced as available for use to all customers.

When attending to the new requirements for enhancements and/or adaptations, all corrections of software problems implemented in the revisions of Version k must be inserted in Version k+1. We call it indirect corrective maintenance.

5 CM³ - Organisational Perspective

CM³ divides the total maintenance process into two main groups, upfront and execution maintenance processes (see Figure 3).

The upfront processes are the Help Desk Process (HDP), and the Product Support Process (PSP). Their combined role is to maximise customer service by attending to all customer demands. A subset of those demands concerns corrective maintenance (software problems). Software problems that cannot be resolved at the upfront level are then transferred on to the maintenance execution processes.
There are two maintenance execution processes. These are the Maintenance Demand Administration Process (MDAP) and the Maintenance Demand Realisation Process (MDRP). The MDAP analyses maintenance demands and decides whether or not they should be attended to. Some of those demands concern software problems. They are managed by the Software Problem Administration Process (SPAP). The SPAP is one of the sub-processes within the MDAP. The other sub-process is called the Software Enhancement Administration Process (SEAP). The SEAP administers software enhancements. Both software problems and enhancements defined in the MDAP process may result in solutions being realised within the MDRP process.

6 Structure of CM³

A corrective maintenance process is seldom monolithic. Instead, it is a complex family of several processes collaborating with each other, where each process performs a clearly defined task. CM³ encompasses some of the most important process models utilised within corrective maintenance. They are:

- **Predelivery/prerelease maintenance process**: Activities to be conducted in order to prepare the software system for future corrective maintenance.
- **Transition/Delivery process**: Activities required for a smooth transfer and delivery of a software system from development to maintenance and customer organisations.
- **Problem Management Process**: Activities required for the reporting, analysing and resolving of software problems [16, 17, 20, 22].
- **Testing Process**: Activities required for testing corrective changes in a software product.
- **Documentation Process**: Activities required for creating a consistent and exhaustive documentation.

### Figure 3. CM³ Roadmap: Organisational perspective [23].

### Figure 4. Structure of CM³.

- **Upfront Maintenance Process**: Activities required for communicating with customers on software problems [23].
- **Maintainer’s Education and Training Process**: Activities required for continuous education and training of maintenance engineers [21].

As depicted in Figure 4, each of the above-mentioned process models has the following structure: (1) *Taxonomy of Activities* listing a set of activities relevant for the process, (2) *Conceptual Model* defining concepts carrying information about the process (3) *Maintenance Elements* explaining and motivating for implementing the maintenance process activities, (4) *Process Phases* structuring some of the CM³ constituent processes into several process phases, (5) *Roles* of individuals executing the process, (6) *Roadmaps* aiding in the navigation throughout the process, and (7) *Maturity Levels* structuring the process activities into three levels: Initial, Defined, and Optimal.

In contrast to other process models, CM³ explains most of its process activities using the CM³: *Maintenance Elements* (see Figure 5). To give maximal guidance to organisations for how to implement or improve corrective maintenance process, each Maintenance Element consist of what-, why- and how-part. The what-part is covered by stating a Maintenance Element, the why-part is covered by the Goal and Motivation motivating for why the Maintenance Element should be implemented, and the how-part is covered by Measures giving general suggestions for how to implement a certain process step.

### Figure 5. Structure of each Maintenance Element.

7 Maturity

Our definition of maturity partly agrees with the CMMI definition [7]. Adapted, however, to the context of corrective maintenance, it reads as follows:
Corrective maintenance process maturity is the ripeness and readiness of the organisation to define, perform, manage, measure, and control their corrective maintenance. It reflects the organisation’s ability (1) to resolve software problems, (2) to consistently apply the corrective maintenance process within the organisation, and (3) to utilise corrective maintenance process as a learning vehicle for improving development and other maintenance processes and for preventing future problems.

CM³ differs from CMM with respect to the designation of maturity levels and placement of its processes within these levels. Each CMM key process area resides at a single maturity level. The primary reason for this decision was to simplify the presentation of how to achieve the next maturity level [7]. Referring to the CM³ model, we have designated three maturity levels (Initial, Defined, Optimal) for each process inherent in the corrective maintenance. We simply do not believe that comprehensive processes such as maintainers’ education and training, testing, problem management and other processes can be sufficiently developed and ripe during one maturity process level.

Due to the varying nature of each CM³ component process, it is difficult to create a common exhaustive definition of each maturity level to be applicable for each component process. The definition of each maturity level must be adapted to the characteristics specific for each process at hand. Generally, the following rules apply:

**Initial:** The process offers little visibility, or gives a distorted view of its status. Usually, this is due to the following reasons: (1) the process is not defined, (2) the process is defined, but not documented, (3) the process is defined and documented, but the documentation is either outdated or inconsistent; hence there is a risk that the process is not consistently adhered to or (4) the process is defined and documented, but not consistently adhered to.

The feedback provided by the process at this level is not always meaningful. Success may depend on the combination of the following factors: (a) competence of the maintenance engineers, (b) their dedicated overtime, (c) low staff turn-over rate, (d) the stability of the applications maintained, (e) the repetitive nature of conducting similar tasks, and (f) assignment of high budget to corrective maintenance.

**Defined:** A simple process model is defined and documented. The process covers the most rudimentary process activities essential for its execution. These activities offer visible milestones for following the progress and for making different kinds of intermediary decisions. Although simple, the process is consistently adhered to.

**Optimal:** At this level, the process provides a fine-grained visibility into its status and progress. We have clear insight into every process step and its results.

Correlation among the CM³ constituent processes performing within corrective maintenance is going to be defined and optimised. For instance, we should not expect that the organisation is on level 3 with respect to testing and problem management, whereas it is still on level 1 with respect to maintainers’ education and training and documentation process.

8 Epilogue

In this paper, we have presented and motivated the Corrective Maintenance Maturity Model (CM³), being presently developed at ABB. CM³ is a fine-grained process model providing maximal visibility into the process. By specialising it to only corrective maintenance, we hope to reduce the uncertainties concerning the definition and scope of corrective maintenance.

Due to the sustained debate on the choice of the name for this type of activity, we became somewhat hesitant whether to use the term “maintenance” for our model. Our uncertainty has forced us to analyse the term “maintenance” from the linguistic point of view [18]. Our analysis has assured us that the term “maintenance” can be used to cover the maintenance categories such as corrective, perfective, adaptive, and preventive. Presently, the maintenance community considers changing the name for this software engineering discipline. We are willing to adapt the name of our model to the new suggestion, if there will come any.

We have considered the international complaints on the choice of the definition of software maintenance and its categories. We have created our own general definition of software maintenance covering three life-cycle phases such as predelivery/prerelease, transition/delivery, and postdelivery/postrelease. We have also slightly modified the wording of the IEEE’s definition of corrective maintenance. For us, corrective maintenance concerns correction of all kinds of defects found in all types of system documentation – not only in software code.

The corrective maintenance process does not live an isolated life within an organisation. It tightly co-operates with other organisational processes. Therefore, it may be difficult to isolate corrective maintenance as a single process within an organisation. Despite this, we feel that it is very important to have a separate process model specialised to only corrective maintenance. How else can we obtain an objective understanding of this domain, and consequently, how else can we contribute to more meaningful and less subjective measurement of the cost of corrective maintenance.

When measuring corrective maintenance, we should not only consider the cost of making changes to the product at the maintenance execution level, but also the cost of communicating on the software problems and their solutions to the customers via the upfront maintenance processes, the cost of predelivery/prerelease and transition/delivery phases, the cost of making changes due
to one and the same problem in several product releases, the cost of travels to the customer’s site, and many other factors.

Our process model may neither be optimal nor exhaustive. However, it is our first step towards scrutinising each maintenance category. We also appeal to the software community world-wide to scrutinise our process model and come up with suggestions for improvements. What could be more valuable than a common world-wide effort of creating and improving a common platform for maintenance. Let us jointly help our “Software Cinderella” to finally stop loosing her shoe(s).

References

[31] Swanson BE, IS Maintainability: Should It Reduce the Maintenance Effort?, SIGCPR 1999, New Orleans LA, USA.