A Model of Maintainability – Suggestion for Future Research

Mira Kajko-Mattsson
DSV, IT University, Sweden
mira@dsv.su.se

Arie van Deursen
CWI and Delft University of Technology, The Netherlands
Arie.van.Deursen@cwi.nl

Rupert Reiger
EADS Deutschland GmbH Corporate Research Center Germany
Rupert.Reiger@eads.net

Gerardo Canfora
The Research Centre on Software Technology (RCOST) Italy
canfora@unisannio.it

Tuomas Ihme
VTT Technical Research Centre, Finland
Tuomas.Ihme@vtt.fi

Torsten Engel
FZI Research Center for Information Technologies Germany,
Tengel@fzi.de

Dan Chiorean
Babes-Boyai University Romania
chiorean@cs.ubbcluj.ro

Meir M Lehman
Department of Computing Imperial College, UK
mml@mdx.ac.uk

Josef Wernke
EADS Eurocopter Germany GmbH
josef.wernke@eurocopter.com

Abstract

Lack of a commonly defined maintainability model hinders us from evaluating and certifying products with respect to maintainability. We cannot compare different products within and across organisations. We have difficulties to evolve and maintain software. We cannot try new development paradigms and evaluate their effects on product maintainability. We have no commonly defined maintainability model to base our software engineering research. In this paper, we outline a suggestion for a model of maintainability. We advocate to the software community to gather its resources in order to develop the model. Such a model would have a substantial long-term impact within research, industry, standardisation authorities and education. This paper lists ideas for a maintainability model with the aim to provide an initial milestone for ample discussion within the software engineering community.

Keywords: *Quality, productivity, product, process, standards, quality metrics and measurement.*

1 Introduction

Many authors have pointed out that the cost of failing to build in maintainability into a software system is very high [22, 28]. Designing for ease of maintenance should already begin when the system is originally conceived [19, 27]. A system ill designed for maintenance cannot have maintainability retrofitted to it later [22]. Increased maintainability, on the other hand, reduces the evolution and maintenance effort thus enabling us to use the same resources to accomplish more change, or accomplish the same change using fewer resources [28].

Unfortunately today, we do not have any useful commonly defined detailed standard for maintainability. Current definitions and models are too general. They do not offer any detailed specification of maintainability [2, 8, 15, 16, 18, 19, 21, 29].

The most detailed quality standard today is ISO 9126 [16]. It defines a set of six quality attributes and provides a framework for quality assessments. One of them is maintainability, defined on a very abstract and general level. At a more detailed level, maintainability is not useful enough and may be viewed differently [3, 30].

Individual attempts have been made by researchers to create models of maintainability [1, 2, 7, 9, 10, 12, 13, 14, 19, 6, 25, 23, 26]. While these models offer interesting insights into various aspects of software quality, they strongly differ from one another. None of them has been strong enough to stimulate significant gains in the quality of software or to gain wide acceptance [10]. All in all, we know what maintainability is, but we have different opinions of what it looks like. There is no universally accepted detailed maintainability model and as a consequence assessing maintainability is next to impossible.

The software community claims that due to the complexity of the notion of maintainability, the organisations should develop their own definitions. What one should do instead is to concentrate efforts on developing common mature production processes. Developing mature processes should help us achieve high quality products.

This opinion has lead to the proliferation of many process-oriented models and standards during the past 20 years. The best known ones are ISO 9000 [17], Capability Maturity Model (CMM) [8] and Spice [11]. They advocate that achieving process maturity equals to achieving product quality; the more mature the process, the higher the quality of the product.

The process maturity trend has resulted in a world-wide movement towards process improvement and certification. Most of the organisations today make their efforts in improving and certifying their processes according to ISO 9000 and/or CMM. They have to do it if they wish to survive within this highly competitive world. The process certification has namely become an important prerequisite for getting new and retaining old customers and subcontractors, and hence it has become an important prerequisite for surviving on the market. **PRODUCT CERTIFICATION, ON THE OTHER HAND, IS ALMOST NON-EXISTENT !**

2 Problems

Today, the software community encounters many problems. Below, we shortly describe them:

• The maintainability concepts are not precisely defined today. Although the term seems self-explanatory, it is not uniformly understood in practice.

• Process maturity is not enough to guarantee the quality of a specific software product. Although a mature process is a reasonable pre-condition for a quality product, we still possess too little knowledge of the relationship between the product and process [4].

• Process evaluation should always be accompanied with product evaluation. Products and processes should be closely linked, and not separated. For this reason, we need a common definition of maintainability based on a common consensus or standard. This would allow us to continuously measure and evaluate software products and provide feedback to the process product improvement. If the product and process measurement do not go hand in hand, then the risk is that you get **an excellent process, but an inferior product.**

• We lack a universally accepted quality and maintainability standard. The only standard we have (ISO 9126) is too general [16]. Its language is not always readily understood. The maintainability attribute is defined on a too coarse-grained level.

• There is also too much focus on assessing and certifying processes. Many of the current models fall short in answering on what happens when a product exceeds the expectation for one maintainability attribute and falls short with another one [22].

• Within research, we do not posses a common basis for conducting basic and strategically important research. This hinders us from (1) creating models/methods providing guidelines for building in and preserving maintainability; (2) creating models/methods for evaluating, assessing and measuring maintainability, (3) understanding the relationship between the product and the process; and (4) restructuring products using re-engineering methods.

• Many software organisations encounter many problems and difficulties such *as (1) difficulties to evolve and* maintain software products; (2) building and maintaining a body of product knowledge; (3) making maintainability comparisons across products and organisations; (4) understanding impact on maintainability when introducing new development paradigms; (5) assuring the quality of the subcontracted, reusable products/ components; and (6) calculating maintenance resources.

3 Outline of the Maintainability Model

In this section, we outline a framework for software maintainability. The framework considers both the product and process aspects. These aspects are presented in Sections 3.1 and 3.2, respectively.

3.1 Product aspects

We suggest create a product maintainability model. We call it *Software Maintainability Model (SMM).* The SMM should apply to traditional software systems as well as to systems developed according to new and emerging approaches. It covers the following parts:

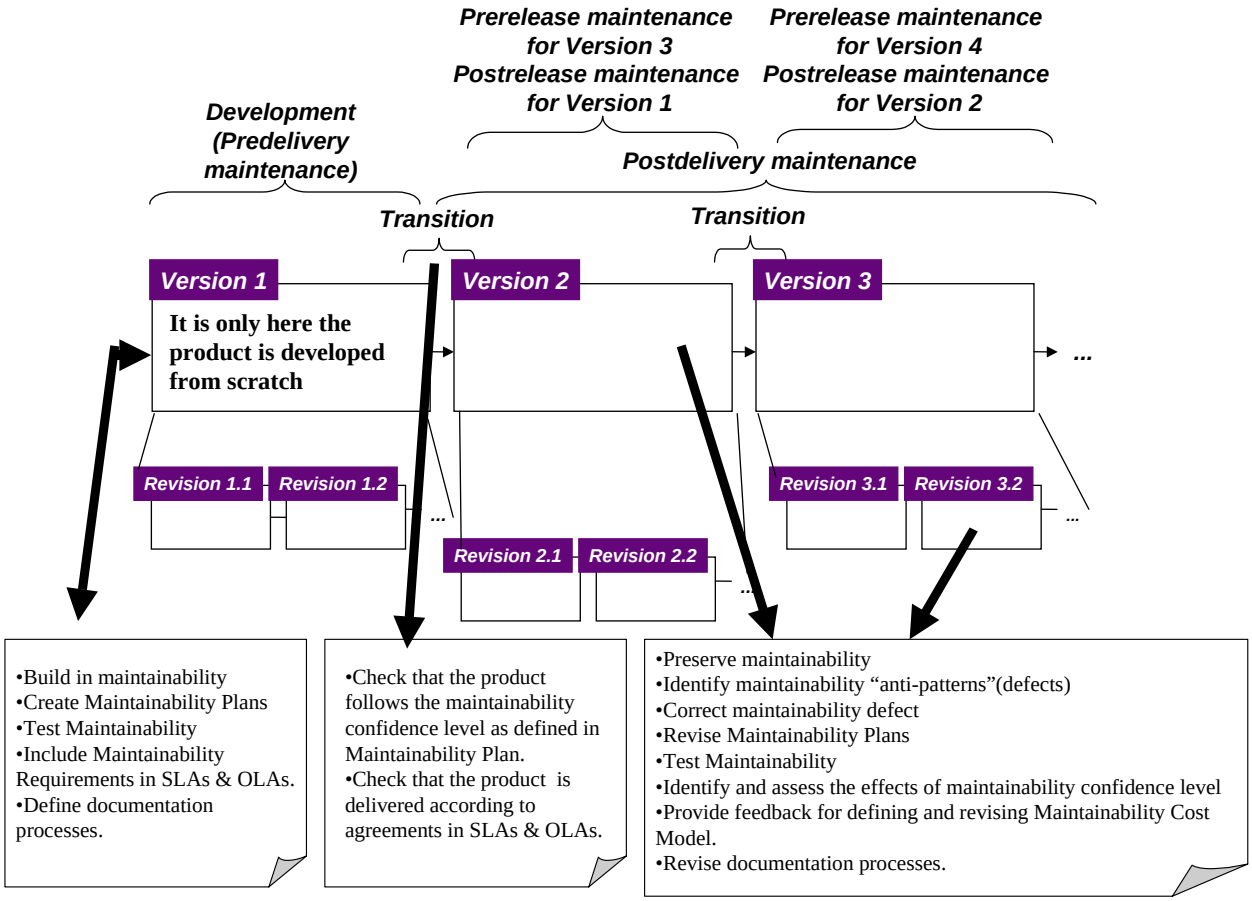

Figure 1. Identifying the role of maintainability within the life-cycle processes

• **Common characteristics**: Irrespective of their differences, many software systems possess the same common fixed characteristics that may be applied throughout the system. Here, the SMM model would identify these common criteria, define, implement, and validate them. These criteria would also apply to different software system types which span from embedded real time systems to web based applications.

• **Variable characteristics**: Some characteristics are not fixed (stable). They may vary with the software types, product types, process or technology used. For instance, the maintainability factor "*understandability*" would have to cover a multitude of structural, modularity and descriptiveness factors that need to be differently satisfied in various methodologies. One must identify these variants, identify the differences amongst them, and suggest how they may be cross-compared, evaluated and calibrated [22].

• **Qualitative and quantitative characteristics**: Some characteristics would be quantitatively measured. Some other characteristics, on the other hand, would have to be assessed qualitatively, for instance, the layout of code. Here, we need to define the qualitative and quantitative models for assessing maintainability characteristics.

• **Traceability characteristics**: A maintainability model would never be complete if we have not complemented it with traceability. The ability of evolving one artefact strongly depends on the possibility to trace it to its previous and successive artefacts, and to the artefacts on lower or higher system levels. Traceability is a pivotal prerequisite for connecting various system levels and for achieving understanding of the system during development, evolution, and maintenance.

• **Maturity levels:** The maintainability characteristics should be assigned to several product maturity levels. These levels would give a degree of confidence in the quality of a software product, and help the organisations improve their product maintainability in a step-wise manner. Hence, the SMM model should be divided into several maturity levels.

• **Taxonomy of maintainability defects**: The list of negative and positive examples (patterns and anti-patterns) would help identify maintainability defects. Within design, for instance, this means that we would study the architecture of design models and try to identify if those design models have in common some bad patterns, referred in the literature as anti-patterns. In order to improve maintainability, it is important to identify and remove these anti-patterns.

• **Samples of maintainable software systems:** Samples of maintainable and unmaintainable software should be provided to aid in product comparisons.

Taking into account that maintainability can be influenced in every software lifecycle phase, all the system document levels, beginning with the enterprise modelling and requirements phase and finishing with the testing one, would be candidates for defining the above-mentioned characteristics.

3.2 Process aspects

A model of product maintainability should not exist in a vacuum. It must be complemented with process models whose common task would be to manage maintainability throughout the whole software life-cycle. We call them Maintainability Process Models (MPM).

As depicted in Figure 1, the role of the MPM models would be to build in, monitor and preserve maintainability into/in software, and systematically ensure that the maintainability-carrying properties are satisfied at all levels of system documents. Please do not mistake these processes for the development and evolution processes which aim at developing and evolving software systems. Here, we mean the processes to be run in parallel with the development and evolution processes, as suggested by the standard ISO/IEC 14764:1999 [19].

Just as the SMM suggests creating models of and metrics for product maintainability, the MPM suggests creating models of and metrics for monitoring various processes and their efficiency to build in or preserve maintainability throughout the product life-cycle.

The SMM models suggests covering three main life-cycle phases: predelivery/prerelease phase, transition phase, and postdelivery phase. They consist of the following models:

- **Maintainability Plan Model**: This model would among many other things, provide a suggestion for a maintainability plan and procedures for building in maintainability into the software system.

- **Maintainability Validation and Certification Process Model**: This model would ensure that the maintainability is built in into the new software components (mainly in enhancements), or that it is preserved in the extant ones (mainly in adaptations and corrections).

- **Maintainability Problem Management Model**: This model would keep a watchful eye on maintainability problems, and take measures to attend to them (correct them) as soon as possible.

- **Maintainability Assessment Model**: This model would help assess the overall maintainability of the product using the product characteristics as defined in the SMM model.

- **Maintainability Cost Model**: This model would measure the cost of spending resources on implementing maintainability and compare it to the benefits gained, or to the cost of negative effects of not implementing maintainability within different life-cycle phases.

- **Process Model for identifying anti-patterns**: This model would define inspection process steps and provide rules and algorithms to be further implemented in a supporting tool supervising that a certain maintainability confidence level has been achieved as defined for each maintainability maturity level.

- **Maintainability maturity levels**: Just as the SMM model, the MPM models would be divided into several maturity levels, where each process level manages a different degree of maintainability confidence. It is important that maintainability confidence would be both reflected in the product and the process. The more mature the process we have, the more maintainability confident product we get.

4 Impact

A commonly defined model of maintainability would substantially contribute to the overall success of the software community. It would have a strong impact on the software industry, academia, and standards. Within the industry, we would achieve the following:

• A commonly defined model of maintainability would substantially improve the well-being and working situation of many software engineers. A maintainable system is a readable, modifiable, extendable, augmentable, testable, traceable, measurable and appraisable system. The effort and frustration of making changes to it would be substantially decreased.

- Software organisations would be able to provide quality guidelines for how to build maintainable systems, and guidelines and metrics for how to evaluate the product quality. This would influence their development, evolution, maintenance and verification processes. The organisations would gain more insight into the products and processes, and their effects on each other. This would in turn improve various types of decision making, lower the costs, shorten time to market and provide more reliable return of investment calculations.

- More maintainable software systems would decrease the cost of customer organisations and the cost of Support Line 1 and Support Line 2 [20].

- A commonly defined model would lead to a better co-operation amongst the subcontractors. It would save their resources to manage different standards. It would facilitate the evaluation of the contribution of the subcontractors.

- In the open-source context, a commonly defined model would assure that the product, managed by many engineers, conforms to some maintainability standard.

- In the product line context, a commonly agreed model, would lead to higher quality, thus making reuse more cost-effective.

- Different vendors would take on responsibility towards maintainability when developing their methodologies, tools and compilers.

- Within research, a standard maintainability model would provide a baseline for various methods, and for communicating research results. Having a solid empirically validated common platform, the global research and its results would be better understood, calibrated and reused [5].

- Finally within standardisation efforts, an empirically validated maintainability model would provide a basis for modifying or extending the current standards, and for developing the new ones.

5 Concluding Remarks

In this paper, we suggest a model of maintainability. It is a shame that so little effort has been spent on this type of empirical research. It should definitely have started much earlier.

We believe that the reason for this is the substantial effort required to construct and empirically validate such a model. The domain of maintainability is too huge to be done by individual researchers or groups of researchers. It requires a combined academic and industrial mobilisation on a world-wide level. Hence, the software community needs to gather its worldwide resources and start acting.

References

[1] Aggarwal K K, et.al., An Integrated Measure of Software Maintainability, Annual Reliability and Maintainability Symposium, 2002, pp. 235-241.

[2] Al-Kilidar, H., Cox, K, Kitchenham, B., The Use and Usefullness of the ISO/IEC9126 Quality Standard, in Proceedings, International Symposium on Empirical Software Engineering, IEEE, 2005.

[3] Andersson T, Eriksson I V, Measuring the Quality Needs of an Organisations's Software, Hawaii Int. Conf, on System Sciences, 1996, pp. 139-154.

[4] Bazzana G, et.al., ISO 9126 and ISO 9000: Friends or Foes? IEEE Software Engineering Standards Symposium, 1993, pp. 79-88.

[5] Biffl S, Grechenig T, Degrees of Consciousness for ReUse of Software in Practice: Maintainability, Balance, Standardization, 7th An. Int. Computer Software and Applications Conference, 1993, pp. 107-114.

[6] McCall J A, Richards P K, Walters G F, Factors in Software Quality", RADC TR-77-369. 1977, Vols, I, II, III, US Rome Air Development Centre Reports NTIS AD/A-049 014, 015, 055, 1977.

[7] Chee C, Power M, Expert Systems Maintainability, Annual Reliability and Maintainability Symposium, IEEE 1990, pp. 415-418.

[8] Capability Maturity Model, Software Engineering Institute, www.sei.org.

[9] Deutsch M, Willis R, Software Quality Engineering, Englewood Cliffs, NJ: Prentice-Hall, 1988.

[10] Dromey R G, A Model for Software Product Quality, IEEE Transactions on Software Engineering, Vol. 21, Iss. 2, February 1995, pp. 146-162.

[11] El Emam K, SPICE , The Theory and Practice of Software Process Improvement and Capability Determination, IEEE Computer Society, 1998.

[12] Franch Z, Carvallo, J P, A Quality-Model –Based Approach for Describing and Evaluating Software Packages, Joint Int. Conf. on Requirements Engineering, 2002, pp. 1-8.

[13] Garcia M, Alvarez J, Maintainability as a Key Factor in Maintenance Productivity: A Case Study, Int. Conference on Software Maintenance, 1996, pp. 87-93.

[14] Software Quality Metrics Methodology Standard, P-106/D20, IEEE Computer Society Press, 1989.

[15] IEEE Standard for Software Maintenance, IEEE Std 1219-1998. The Institute of Electrical and Electronics Engineers, Inc. 1998.

[16] ISO/IEC 9126 International Standard ISO/IEC 9126: Software Engineering - Product Quality.

[17] ISO/TCI76/SC2/WG17/N12, ISO/DIS 9000-3, Quality Management and Quality Assurance – Part 3: Guidelines for the applications of ISO 9001:1994 to the design, development, supply, installation and maintenance of computer software.

[18] ISO/IEC 12207 International Standard, Information technology – Software life cycle processes, Ref. Nr. ISO/IEC 12207:1995(E).

[19] ISO/IEC 14764: Information technology – Software Maintenance, Ref. No. ISO/IEC 14764:1999(E).

[20] Kajko-Mattsson, M., *Evaluation of CM^3: Front-End Problem Management within Industry*, In Proceedings, IEEE The 10th Conference on Software Reengineering and Maintenance (CSMR 2006), IEEE Computer Society Press: Los Alamitos, CA, 2006.

[21] Kitchenham B, Towards a Constructive Quality Model, Software Engineering Journal, July 1987, pp. 105-112.

[22] Koscianski A, Candido Bracarense Costa J, Combining Analytical Hierarchical Analysis with ISO/IEC 9126 for a Complete Quality Evaluation Framework, International Symposium and Forum on Software Engineering Standards, 1999, pp. 218-226.

[23] Nance , Software Quality Indicators: A Holistic Approach to Measurement, 4th Annual Software Quality Workshop, Alexandria Bay, New York, Aug. 1992.

[24] Martin J, McClure C, Software Maintenance, The Problem and Its Solutions, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1983.

[25] Muthana S, et.al., A Maintainability Model for Industrial Software Systems Using Design Level Metrics, IEEE, 2000, pp. 248-256.

[26] Oman P, Hagemeister J, Metrics for Assessing Software System Maintainability, In Proceedings, International Conference on Software Maintenance, IEEE, Computer Society Press in Los Alamitos CA, 1992, pp. 337-344.

[27] Pigoski TM, Practical Software Maintenance, John Wiley & Sons, 1997.

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

[29] The TickIT Guide, A Guide to Software Quality Management System Construction and Certification to ISO 9001, DISC TickIT Office, 1998.

[30] Vollman T E, Software Quality Assessments and Standards, Computer, Vol. 26, Issue 6, June 1993, pp. 118-120.