Journal of Theoretical and Applied Information Technology
10th November 2013. Vol. 57 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A COMPREHENSIVE STUDY OF CMMI BASED FRAMEWORK FOR COLLABORATIVE SOFTWARE MAINTENANCE

[1]HANEEN AL-AHMAD, [2]RODZIAH ATAN, [3]ABDUL AZIM ABD GHANI, [4] MASRAH AZMI MURAD

Faculty of Computer Science & IT, University Putra Malaysia
43400 UPM, Serdang, Selangor, Malaysia
Haneeen_0005@hotmail.com & (rodziah, azmi & masrah)@fsktm.upm.edu.my
Faculty of Computer Science & IT, Information System Department, University Putra Malaysia, 43400 UPM, Serdang, Selangor, Malaysia

ABSTRACT

Software maintenance (SM) environment is a highly complex area, knowledge-driven and collaborative. Therefore, Capability Maturity Model Integration (CMMI) is a process improvement approach that provides organizations with the essential elements of effective processes that ultimately improve their performance. We propose a new framework of CMMI based on Multi-Agent System (MAS) to identify the process measurement of SM. The proposed MAS architecture includes three types of agents: Personal Agent (PA), Maintenance Agent (MA) and Key Process Area Agent (KPAA). In order to verify our proposed CMMI framework based on MAS architecture, a pilot study is conducted using a questionnaire survey. Rasch Model is used to analyze the pilot data. Item reliability is found to be poor and a few respondents and items are identified as misfits with distorted measurements.

Keywords: *Capability Maturity Model Integration, Software Maintenance, Software Maintenance Process, Multi Agent System and Rasch Model*

1. INTRODUCTION

Knowledge transfer of a large number of the best practices described in a maturity model has proved difficult [1]. This is especially true during the training of an assessor or a new participant in a process improvement activity. Software measurement, in order to be effective, must be focused on specific goals; applied to all life-cycle products, process and resources; and interpreted based on characterization and understanding of the organizational context, environment and goals [2]. Software maintenance (SM), according to IEEE definition, is a modification of a software product after delivery in order to correct faults, to improve performance or other attributes, to adapt a product to a changed environment, or to improve the product maintainability [3]. A maturity level is a well-defined evolutionary plateau toward achieving a mature software process. Each maturity level provides a layer in the foundation for continuous process improvement. In CMMI models with a staged representation, there are five maturity levels [4]. Initial, Managed, Defined, Quantitatively Managed and Optimizing as illustrated in table 1.

Table 1: CMMI Staged Representation- Maturity Levels

Level	Continuous Representation Capability Levels	Staged Representation Maturity Levels
Level 1	Performed	Initial
Level 2	Managed	Managed
Level 3	Defined	Defined
Level 4	Quantitatively Managed	Quantitatively Managed
Level 5	Optimizing	Optimizing

Maturity levels consist of a predefined set of process areas. The maturity levels are measured by the achievement of the specific and generic goals that apply to each predefined set of process areas. The following sections describe the characteristics of each maturity level [5].

At maturity level 1 (Initial Level), processes are usually ad hoc and chaotic. The organization

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

usually does not provide a stable environment. Success in these organizations depends on the competence and heroics of the people in the organization and not on the use of proven processes. Maturity level 1 organizations often produce products and services that work; however, they frequently exceed the budget and schedule of their projects. Maturity level 1 organizations are characterized by a tendency to over commit, abandon processes in the time of crisis, and not be able to repeat their past successes.

At maturity level 2 (Managed Level), an organization has achieved all the specific and generic goals of the maturity level 2 process areas. In other words, the projects of the organization have ensured that the requirements are managed and that processes are planned, performed, measured, and controlled [4].

At maturity level 3 (Defined Level), an organization has achieved all the specific and generic goals of the process areas assigned to maturity levels 2 and 3. At maturity level 3, processes are well characterized and understood, and are described in standards, procedures, tools, and methods.

At maturity level 4 (Quantitatively Managed Level), an organization has achieved all the specific goals of the process areas assigned to maturity levels 2, 3, and 4 and the generic goals assigned to maturity levels 2 and 3. At maturity level 4 Sub-processes are selected that significantly contribute to overall process performance. These selected sub-processes are controlled using statistical and other quantitative techniques.

At maturity level 5 (Optimizing Level), an organization has achieved all the specific goals of the process areas assigned to maturity levels 2, 3, 4, and 5 and the generic goals assigned to maturity levels 2 and 3. Processes are continually improved based on a quantitative understanding of the common causes of variation inherent in processes. Maturity level 5 focuses on continually improving process performance through both incremental and innovative technological improvements [6].

Multi Agent System (MAS) has attracted a great deal of attention in recent years because they have introduced a new paradigm for analyzing, designing, and implementing software systems. A lot of multi-agent methodologies have been born and improved since the presence of MAS. They have shown a great power in solving problems. MAS is designed and implemented as several interacting agents. MAS are ideally suited to representing problems that have multiple problem solving methods and multiple perspectives. MAS takes initiative where appropriate, and socially interact, where appropriate, with other artificial agents and humans in order to complete their own problem solving and to help others with their activities [7],[8].

2. LITERATURE REVIEW

Software maintenance function suffers from a scarcity of management models that would facilitate its evaluation, management and continuous improvement. This paper is part of a series of papers that presents a Software Maintenance Capability Maturity Model (SMCMM). The contributions of this specific paper are: 1) to describe the key references of software maintenance; 2) to present the model update process conducted during 2003; and 3) to present, for the first time, the updated architecture of the model [9].

SM process is one of the most costly activities within information system practice. The purpose of this paper is to address some of the difficulties in this process, by proposing a framework for the development of maintenance mode. Essential to the software maintenance process is an ability to understand not only the software but the required changes as well. This can only be achieved where the relevant knowledge is available. Based upon this primary requirement, the proposed framework has made the knowledge as its basis for modeling other requirements for software maintenance model development. The framework first identifies the three operational elements, i.e. Function, static entity and dynamic entity, required for the general software maintenance process. With respect to the knowledge (as part of the dynamic entity components), the framework shows how these three operational elements should behave and interact amongst themselves to deliver a successful software maintenance model [10].

Holgeid [11], presents the main results from a survey investigation performed in Norwegian organizations within the area of software development and maintenance. The results are based on responses from 53 Norwegian organizations. Somewhat surprisingly, the amounts of both traditional and functional maintenance work are significantly higher than in the similar investigation done five years earlier. It is also significantly higher than in the USA and in other countries. Also too much of the scarce IT-personnel spent their time on tasks that do not add value to the users of the systems.

[12] presents an overview of the measurement practices that are being introduced for level 3 and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

higher to the Software Maintenance Maturity Model (S3M). Software maintenance still does not receive a noticeable share of management attention and suffers from lack of planning, as often illustrated by its crisis management style. Part of the problem is that maintenance is typically perceived as being expensive and ineffective. Moreover, few proposals of best practices have been put forward which can readily be applied in industry. In general, the software engineering community expects that product quality will be enhanced if the maintenance process is improved.

[13] Deals with a method developed for software maintenance called Remote Maintenance Shell. It allows software installation, modification and verification on the remote target system without suspending its regular operation. The method is based on remote operations performed by mobile agents. The role of Remote Maintenance Shell in software maintenance is elaborated, as well as its architecture. A case study on version replacement of an object-oriented application is included.

[16] Presented results of introducing an agile process based on extreme programming, XP, in an evolutionary and maintenance software development environment. The agile process was introduced to a large software development organization. The process was applied by a team during eight months. The conclusions indicate that it in this case is more difficult to introduce XP, in its original appearance, into the case environment than in less complex environments. The complexity of the organization made it necessary to redesign many of the practices in order for them to fit the needs of the software development team.

3. METHODOLOGY

In this phase, three main activities were undertaken. First, the generic CMMI, MAS and SM models are studied and summarized. Then the components are extracted from the above generic models. Afterward, these components are revised to remove redundant and non-related components.

A survey was conducted in selected 41 respondents from three organizations participated in the survey. Fifty questionnaires were distributed to the respondents, and only forty one questionnaires were returned. The questionnaire data were verified and was analyzed using Rasch Model. The result of the survey contributed to the formulation of the proposed framework.

A suitable questionnaire set is then developed to verify the initial components. The main aim of the questionnaire is to determine which components are deemed important by SM practitioners. The questionnaire is developed using a 4-Likert Scale order of importance, with ’1’ being less important and ‘4‘denotes more important. To ensure reasonable face validity, questionnaires were further deliberated and refined by academic lectures in the Software Engineering field, a statistician and three SM managers.

To verify the constructs of the questionnaire, a pilot group of respondents shall test the questionnaire. The pilot group respondents are selected based on convenience sampling from an in-house SM organization. The Rasch rating scale model is used in the analysis to determine if the questions are well understood by respondents (i.e. Hard to answer questions), or if the questions are mundane and trivial (i.e. Too easy) that perhaps the questions could just be left out. Rasch is a probabilistic model that uses logit as the measurement units, by transforming the ordinal data and raw scores into a linear scale [17]. Being linear, research enables us to conduct more constructive analyses, such as to determine the reliability of respondents and items, determine outliers for both respondents and items that do not fit the model to enable us to investigate further to explain the inconsistency. This form of validity comes from the fit of the observed person-item responses to a useful definition of measurement of the estimated values of item and person measures [18]. As a result, some problematic questionnaire items could be identified, revised and some discarded to make the questionnaire more acceptable. The revised questionnaire shall later be used for further analysis to verify important components for CMMI based on the MAS model for collaborative SM.

4. PROPOSED FRAMEWORK

Currently, there is a lack of formal CMMI based MAS models for the collaborative SMP environment, and there are no hard and fast rules on how to formulate a CMMI based MAS model. April, Abran & Dumke reviewed and synthesized several CMMI models into a comprehensive generic collaborative SM model. However, how the synthesized CMMI based MAS model was synthesized from components from related generic CMMI models and existing SMP models. The main question is – are all these synthesized components Combined and validated was not discussed. Using a similar approach, a CMMI based MAS model for the collaborative SM environment shall be

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

important to the users, developers and maintainers in SMP environments? To validate the initial model, [9],[16],[19],[20], offers insights on developing and validating the instrument construct to select the important components, respectively. Figure 1 shows a schematic representation of the framework. The framework has been built by using four layers.

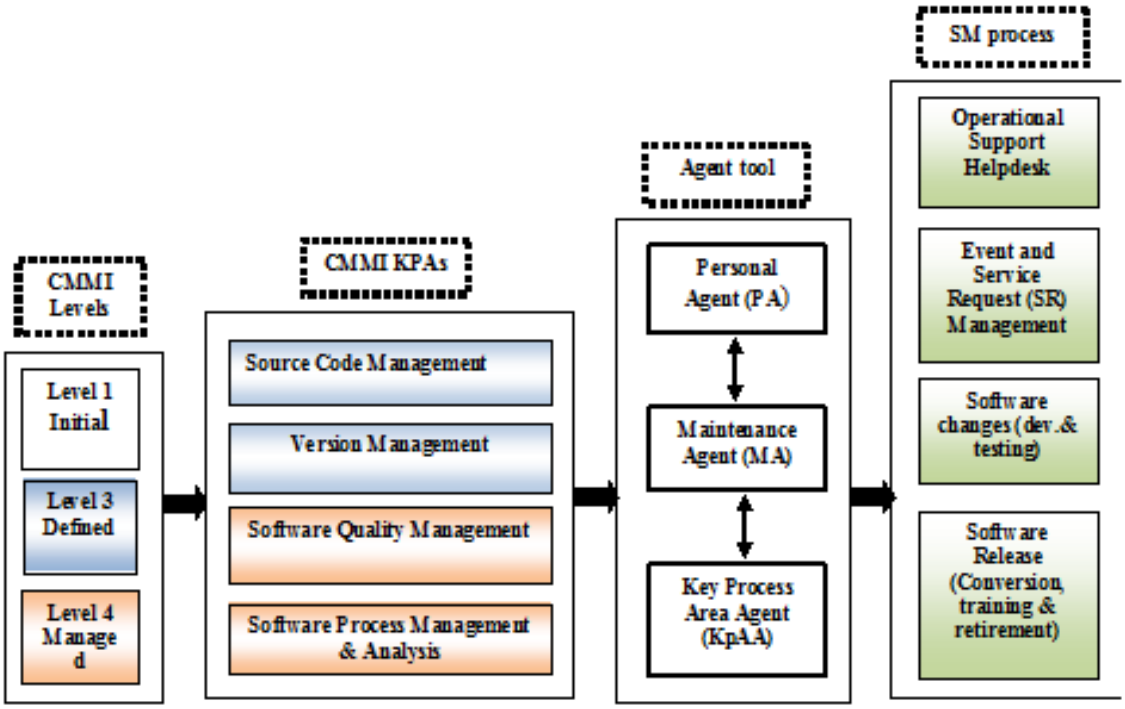

Figure 1. CMMI Based on MAS Framework

Model's Goals: (1) to measure the quality of SMPs. (2) to evaluate the continuous improvement of the SMPs. (3) to generate and facilitate the source code of the SMPs improvement that could be improves by the developers and maintainers.

KPAs' Goals: a) to ensure that events and service requests (SRs) are identified and registered daily; b) to determine the relative importance, within the current workload, of new events and SRs; and c) to ensure that the workload is focused on approved priorities.

Scope of the model: Models are often an abstract representation of reality. For a better mapping with the maintainers and reality, the proposed model includes many of the essential perspectives of the software maintainer, and as much as possible of the maintainer's practical work context. Our model is intended to describe specific techniques or all the technologies used by maintainers. Our model has recently adopted from the continuous representation of CMMI and only deal with the levels 0 (absent), 3 & 4 of the CMMI because the evaluation and the measurement of the quality of the SMPs only existing in these levels.

5. RESULTS AND DISCUSSION

The pilot data were tabulated and analyzed using WinSteps, a Rasch tool. The results of the Person and Item summary statistics and measures are tabulated in Table 1 and Table 2. The results of the survey are analyzed in three parts; data reliability, fitness of respondent and items data and determination of component groups cutoff points.

5.1 Data Reliability

Summary statistics for respondents (person) and items (questions) are depicted in Table 1 and Table 2, respectively. 41 respondents returned the survey questionnaire. Out of which, Rasch identified an extreme score which will later be excluded from further analysis.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1. Summary of Measured (Non-Extreme) Persons

	Raw Score	Count	Measure	Model Error	Infit		Outfit	
					MNSQ	ZSTD	MNSQ	ZSTD
MEAN	133.8	42.8	0.49	0.27	1.02	-0.2	1.01	-0.2
S.D.	14.9	3.5	0.69	0.02	0.52	2.1	0.53	2
MAX.	167	45	2.64	0.34	3.14	6.4	3.37	6.7
MIN.	86	30	-0.65	0.25	0.28	-4.5	0.28	-4.4

Real RMSE .30 Adj.SD .62 Separation 2.10 Person Reliability .82
Model RMSE .27 Adj.SD .64 Separation 2.35 Person Reliability .85
S.E. Of Person Mean = .11
Maximum Extreme Score: 1 Persons
Valid Responses: 95.0%
Person Raw Score-To-Measure Correlation = .51 (approximate due to missing data)
CRONBACH ALPHA (KR-20) Person Raw Score Reliability = .94 (approximate due to missing data)

Table 2. Summary of Measured Items

	Raw Score	Count	Measure	Model Error	Infit		Outfit	
					MNSQ	ZSTD	MNSQ	ZSTD
MEAN	119.8	38.3	0.02	0.3	1	0	1	0.1
S.D.	16.7	3.2	0.64	0.08	0.12	0.6	0.15	0.7
MAX.	150	40	1.16	0.6	1.29	1.5	1.4	1.9
MIN.	88	29	-1.2	0.2	0.83	-1.3	0.74	-1.3

Real RMSE .32 Adj.SD .54 Separation 1.69 Item Reliability .74
Model RMSE .27 Adj.SD .64 Separation 2.35 Item Reliability .75
S.E. Of Person Mean = .09

The spread of person responses is = 3.29 logit is fair. This is due to extreme responses by a participant. However, Reliability = 0.82 and Cronbach Alpha=0. 94 indicates high reliability data and hence the data could be used for further analyses.

On the questionnaire items, the summary of 45 measured questionnaire items (see Table 4.3) reveals that the spread of data at 2.36 logit and reliability of 0.74 are good and fair, respectively. Details on measured items are listed in Table 4.4. The acceptable limits are 0.4 < Acceptable Point Measure Correlation < 0.8 and 0.5 < Outfit Mean Square < 1.5, and -2.0 < Outfit z-standardized value < 2.0).

6. CONCLUSION

The CMMI based on MAS Framework components for the collaborative SM environment was initially synthesized from the generic CMMI, MAS and SM frameworks. A questionnaire survey followed by the expert opinion survey was conducted to ascertain the important components of the framework. The CMMI based on MAS framework consists of Knowledge Required for SM Activities, SM Governance Tools, CMMI Tools and Agent Tools. To formulate the CMMI based on the MAS framework for collaborative SM, the components of CMMI tools, SM governance tools, and agent tools are compiled from various literatures. An initial model of modified CMMI based on MAS components for collaborative SM is proposed. The relationships between these components are used to construct the questionnaire, which were tested in a pilot study. RUMM was used in analyzing pilot questionnaire. Item reliability is found to be poor and a few respondents and items were identified as misfits with distorted measurements. Some problematic questions are revised and some predictably easy questions are excluded from the questionnaire.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFERENCE

[1] Abran, A. Moore, J. Bourque, W. Dupuis, P and Tripp, L.. Guide for the Software Engineering Body of Knowledge (SWEBOK), Ironman version, IEEE Computer Society Press: Los Alamitos CA, 2004. 6(1)- Pp: 6-15.

[2] Basili, V.R. Caldiera, G and Rombach, H.D. (1994). The Goal Question Metric Approach. In Encyclopedia of Software Engineering, Wiley, pp. 528-532.

[3] Pigoski, T.M. Practical Software maintenance: Best Practice for Managing your Software Investment, 1st edi. 1997. Wiley.

[4] CMMI Maturity Levels, available at: http://www.tutorialspoint.com/cmmi/cmmi-maturity-levels.htm 2002. (Accessed on July 2013).

[5] April, A., Huffman Hayes, J., Abran, A., and Dumke, R. Software Maintenance Maturity Model (SMmm): the software maintenance process model', Journal of Software Maintenance and Evolution: Research and Practice, 2005. 17 (3), pp. 197-223.

[6] April, A., Desharnais, J.M., and Dumke, R. A Formalism of ontology to support a software maintenance knowledge-based system. A Formalism of ontology to support a software maintenance knowledge-based system. 2006. Pp. 331-336.

[7] Talib, A. M., Atan, R., Abdullah, R., and Murad, M. A. A. Towards New Data Access Control Technique Based on Multi Agent System Architecture for Cloud Computing." Communications in Computer and Information Science 189 CCIS (Part II), 2011a. Pp. 268-279.

[8] Talib, A. M., Atan, R., Abdullah, R., and Murad, M. A. A. CloudZone: Towards an Integrity Layer of Cloud Data Storage Based on Multi Agent System Architecture. ICOS, 2011b. Pp.127-132.

[9] April, A., Abran, A., and Dumke, R.R. 'SMCMM model to evaluate and improve the quality of the software maintenance process: SMCMM model to evaluate and improve the quality of the software maintenance process' (IEEE, 2004, edn.), 2004, 1st Edi. pp. 243-248.

[10] Deraman, A.. 'A Framework For Software Maintenance Model Development', Malaysian Journal of Computer Science, 1998. 11(2), Pp. 23-31.

[11] Holgeid, K.K., Krogstie, J., and Sjberg, D.I.K.. 'A study of development and maintenance in Norway: assessing the efficiency of information systems support using functional maintenance', Information and Software Technology, 2000, 42, (10), Pp. 687-700.

[12] April, A., and Abran, A. 'A software maintenance maturity model (S3M): Measurement practices at maturity levels 3 and 4', Electronic Notes in Theoretical Computer Science, 2009. 233, pp. 73-87

[13] Lovrek, I., Jezic, G., Kusek, M., Ljubi, I., Caric, A., Huljenic, D., Desic, S., and Labor, O. 'Improving software maintenance by using agent-based remote maintenance shell', in Editor (Ed.)^(Eds.): 'Book Improving software maintenance by using agent-based remote maintenance shell' (IEEE, 2003, edn.), 2003. Pp. 440-449.

[14] Svensson, H., and Host, M. 'Introducing an agile process in a software maintenance and evolution organization: Introducing an agile process in a software maintenance and evolution organization' (IEEE, 2005, edn.), 2nd edi. 2005. Pp. 256-264.

[15] Bond, T. G., and Fox, C. M. Applying the Rasch Model: Fundamental Measurement in the Human Sciences. Journal of Educational Measurement, 2003. Volume 40, pp: 185-187.

[16] Wright, B. D., and Stone, M. H. Measurement Essentials. Delaware: 1999. Wide Range.

[17] Chen, J.C., and Huang, S.J. 'An empirical analysis of the impact of software development problem factors on software maintainability', Journal of Systems and Software, 2009. 82, (6), pp. 981-992.

[18] Jung, H.W., and Goldenson, D.R. (). 'Evaluating the relationship between process improvement and schedule deviation in software maintenance', Information and Software Technology, 2009 51(2), pp. 351-361.