SOFTWARE MAINTAINABILITY ASSESSMENT BASED ON COLLABORATIVE CMMI MODEL

Haneen Al-Ahmad[1], Rodziah Atan[1], Abdul Azim Abd Ghani[1] and Masrah Azmi Murad[1]

[1]*University Putra Malaysia, Malaysia*

ABSTRACT

Software constantly needs new features or bug fixes. Maintainable software is simple to extend and fix which encourages the software's uptake and use. The Software Sustainability Institute can advise you on the design and development of maintainable software that will benefit both you and your users. Therefore, capability maturity model integration (CMMI) is a process improvement approach that provides organisations with the essential elements of effective processes that ultimately improve their performance. The propose maintainability assessment of cmmi based on multi-agent system (MAS) to identify the processes measurement of SM. in order to verify our proposed CMMI framework based on MAS architecture, pilot study is conducted using a questionnaire survey. Rasch model is used to analyse the pilot data. Items reliability is found strong correlation between measured and the model designed. The results shows that the person raw score-to-measure correlation is 0.51 (approximate due to missing data) and Cronbach Alpha (kr-20) person raw score reliability = .94.

Keywords:

Capability Maturity Model Integration, Software Maintenance, Software Maintenance Process, Multi Agent System and Rasch Model.

INTRODUCTION

Knowledge transfer of a large number of the best practices described in a maturity model has proved difficult (Abran et al., 2004). 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 characterisation and understanding of the organisational context, environment and goals (Basili et al., 1994). Software maintenance (SM), according to IEEE definition, is a modification of 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 (Pigoski, 1997). A maturity level is a well-defined evolutionary Figureau 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 (CMMI Maturity Levels, 2002): Initial, Managed, Defined, Quantitatively Managed and Optimizing as illustrated in Table 1.

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 April et al.

DOI:10.63646/j.ijirm.2014.020107

At maturity level 1 (Initial Level), processes are usually ad hoc and chaotic. The organisation usually does not provide a stable environment. Success in these organisations depends on the competence and heroics of the people in the organisation and not on the use of proven processes. Maturity level 1 organisations often produce products and services that work; however, they frequently exceed the budget and schedule of their projects. Maturity level 1 organisations are characterised 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 organisation has achieved all the specific and generic goals of the maturity level 2 process areas. In other words, the projects of the organisation have ensured that requirements are managed and that processes are planned, performed, measured, and controlled (CMMI Maturity Levels, 2002).

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

At maturity level 3 (Defined Level), an organisation 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 characterised and understood, and are described in standards, procedures, tools and methods.

At maturity level 4 (Quantitatively Managed Level), an organisation 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 organisation 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 (April et al., 2006).

Multi Agent System (MAS) has attracted a great deal of attention in recent years because they have introduced a new paradigm for analysing, 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 take 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 (Talib et al, 2011a; Talib et al, 2011b).

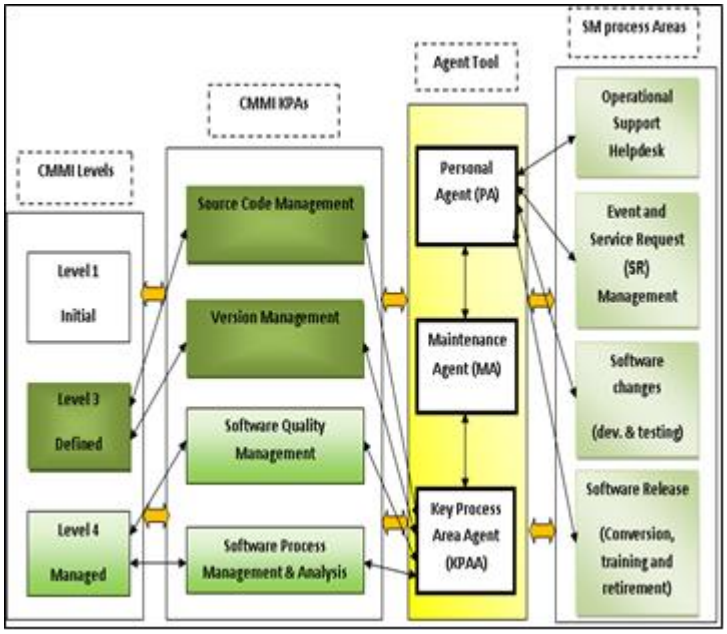

Figure 1: Research Methodology Framework

The methodology of this research as shown in Figure 1 is based on enhancement of the quality of software maintainability evaluation model via CMMI process which enables a better combining of multi-agent procedure in order to have a successful software maintainability system. The steps taken in this research starts with the identification of software maintenance (SM) key process areas (KPAs) in each of CMMI level. The KPAs identified are validated by industry expert in SM domain. Next step proposed by this research is aiding the identified KPAs for specific CMMI level with MAS agents. There are three agents developed to cope with the KPAs indicated. Results from the MAS execution for software maintenance key process areas in CMMI are then mapped to categories of SM activities. This research has categorised four major SM activities, tested them with MAS agents to prove that the quality has increased and SM activities accomplishment duration reduced.

There are four phases of the study in the form of a flow chart diagram, as specified method is needed to give details of the research flow based on collecting information, problem

statement, objectives of the research and observations of the theory as stated in the literature review.

SOFTWARE MAINTENANCE FUNCTION

Software maintenance (SM) 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 (April et al., 2004).

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 model. 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 modelling 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 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 (Deraman, 1998).

Holgeid et al. presents the main results from a survey investigation performed in Norwegian organisations within the area of software development and maintenance. The results are based on responses from 53 Norwegian organisations. 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 for the users of the systems.

April et al. presents an overview of the measurement practices that are being introduced for level 3 and 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.

Lovrek et al. 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.

Svensson and Host 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 organisation. The process was applied by a team during eight months. The conclusion indicated that it in this case is more difficult to introduce XP, in its original appearance, to the case environment than to less

complex environments. The complexity of the organisation made it necessary to redesign many of the practices in order for them to fit the needs of the software development team.

RESULTS AND DISCUSSION

The pilot data were tabulated and analysed using Win Steps, a Rasch tool. The statistics and measures are tabulated in Figure 2 of the comparative study, the results summary shows of the questionnaire and model shows that the between the model and questionnaire outputs are highly correlated as shown in Figure 2. The results of the survey are analysed in three parts; data reliability, fitness of respondent and items data and determination of component groups cut-off points.

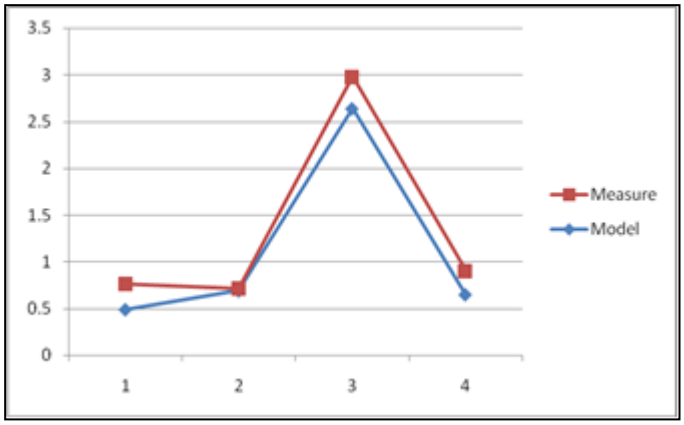

Figure 2: Data Correlation between Measured and Model

Data Reliability: Summary statistics for respondents (Questionnaire) and created model are depicted in Figure 3 respondents returned the survey questionnaire accordingly and compared with the model designed using Rasch software. Out of which, Rasch identified a significant correlation. The standard deviation were measured based on the observation from the questionnaire.

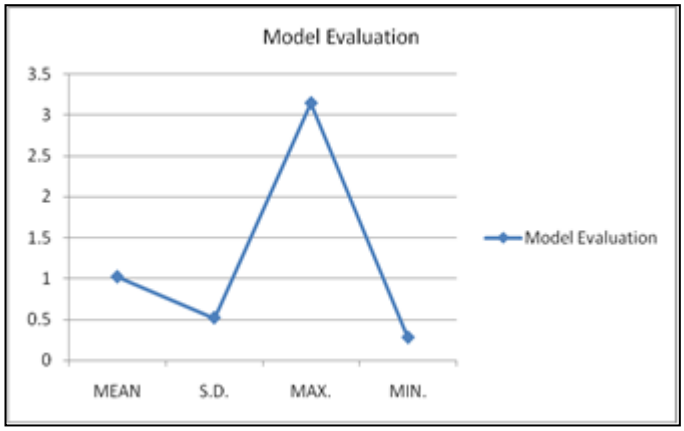

Figure 3: Model Evaluation

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 reliable data and hence the data could be used for further analyses. In the questionnaire items, the summary of 45 measured questionnaire items (see Table 2) 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 1. 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-standardised value < 2.0) as shown in Table 2 below.

Table 2: Summary of Measured Items

	Raw Score	Count	Measure	Model Error	Infit MNSQ	Infit ZSTD	Outfit MNSQ	Outfit 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 0.32. Adj. SD 0.54. Separation 1.69
Item Reliability 0 .74. Model RMSE 0.27
Adj.SD S.E. of Person Mean = .09.

CONCLUSION

The CMMI based on MAS Framework components for collaborative SM environment was initially synthesised from the generic CMMI, MAS and SM frameworks. A questionnaire survey followed by expert opinion survey was conducted to ascertain the important components for 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 MAS framework for collaborative SM, the components on 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 analysing 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.

REFERENCES

[1] Abran, A., Moore, J., Bourque, W., Dupuis, P. & Tripp, L. (2004). Guide for the software engineering body of knowledge (SWEBOK), Ironman version, IEEE Computer Society Press: Los Alamitos CA, 6-1-6-15.

[2] April, A., Desharnais, J. M., & Dumke, R. (2006). A Formalism of ontology to support a software maintenance knowledge-based system. in Editor: Book a formalism of ontology to support a software maintenance knowledge-based system, 331-336.

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

[4] April, A., Abran, A., & Dumke, R.R. (2004). SMCMM model to evaluate and improve the quality of the software maintenance process. in Editor: Book SMCMM model to evaluate and improve the quality of the software maintenance process. IEEE, 243-248.

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

[6] Basili, V.R. Caldiera, G & Rombach, H.D. (1994). The goal question metric approach. Encyclopedia of Software Engineering, Wiley, 528-532.

[7] Bond, T. G., & Fox, C. M. (2003). Applying the Rasch Model: Fundamental measurement in the human sciences. *Journal of Educational Measurement, 40,* 185-187.

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

[9] CMMI Maturity Levels. (2002). available at: http://www.tutorialspoint.com/cmmi/cmmi-maturity-levels.htm (Accessed on July 2012).

[10] Deraman, A. (1998). A framework for software maintenance model development. *Malaysian Journal of Computer Science, 11 (2),* 23-31.

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

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

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

[14] Pigoski, T. M. (1997). Practical software maintenance: Best practice for managing your software investment, Wiley.
[15] Svensson, H., & Host, M. (2005). Introducing an agile process in a software maintenance and evolution in organisation. in Editor: Book Introducing an agile process in a software maintenance and evolution organisation. IEEE, 256-264.
[16] Talib, A. M., Atan, R., Abdullah, R., & Murad, M. A. A. (2011a). 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), 268-279.
[17] Talib, A. M., Atan, R., Abdullah, R., & Murad, M. A. A. (2011b). CloudZone: Towards an integrity layer of cloud data storage based on multi agent system architecture. ICOS, 127-132.
[18] Wright, B. D., & Stone, M. H. (1999). Measurement essentials. Delaware: Wide Range
[19] Yeh, D., Whu, C. K., & Jeng, J. H. 2006. 'Software maintenance in Taiwan and a comparative analysis with other countries', *System, 111 (64)*, 47.