CAPABILITY AND MATURITY. CHARACTERISTICS USED IN SOFTWARE RELIABILITY ENGINEERING MODELING

Marian Pompiliu CRISTESCU

“Lucian Blaga” University, Sibiu, Romania
marian.cristescu@ulbsibiu.ro

Julian Andreev VASILEV

University of Economics, Varna, Bulgaria
vasiliev@ue-varna.bg

Miglena Velichkova STOYANOVA

University of Economics, Varna, Bulgaria
m_stoyanova@ue-varna.bg

Ana-Maria Ramona STANCU

“Spiru Haret” University of Bucharest, Romania
ana_maria_ramona@yahoo.com

ABSTRACT

This paper presents how to use the Capability Maturity Model Integration (CMMI) in software engineering modeling processes. Any variation of CMMI, depending on the scope of application, can also be used to assess the process maturity of an organization. Regarding the Software Reliability Engineering (SRE) process, the activities in the SRE process have been briefly described, noting that the application of SRE in all software-based products leads to good control over the development process.

KEYWORDS: capability, maturity, model, software engineering

JEL Codes: C31, C32

1. Introduction

The size and complexity of software systems have grown impressively over the last decades, and this trend will certainly continue in the future. Industry data shows that software sizes for various systems and applications have increased exponentially over the last 40 years (Humphrey, 2009). The trend of such growth in the telecommunications, business, defense and transport industries shows a tenfold growth rate every five years. Due to this growing dependence, software failures can lead to

serious, even fatal, consequences in critical safety systems, but also in normal business. Over time, “*software failures can lead to serious, even fatal, consequences in safety-critical systems as well as in normal business*” (Lyu, 1996).

Software is a thing without a physical representation, which gives it both advantages and disadvantages (Bankov, 2019). Advantages of software refer to the fact that most of the software tools that people use provide an increased level of satisfaction due to the correct way of

functioning due to the modern methods and techniques of design, development, implementation and maintenance of these software tools. The disadvantages are related to the occurrence of these software tools failures caused by the occurrence, totally random errors and/or failures. The software crisis, identified in the early 1960's, is still stubborn to remain, and "software engineering" has not yet fully evolved into a real engineering discipline. The unstable software crisis exposes colossal opportunities for both researchers and software practitioners. The evolution of the company has led to the need for software engineering practice to create and maintain software systems with high quality indexes and features. Such an approach produces a significant increase in the efficiency of software design/development/implementation/maintenance processes at the level of companies that use modern software design and development techniques and methods compared to those that are late to implement of methods in the software development process.

Being a major task in the error/failure forecasting process, the software reliability modeling has attracted much research attention on the one hand for the software reliability estimation process, and on the other hand to make predictions about the level of reliability the software will achieve. In software reliability modeling, tools (models) are used that allow for random evaluation of this quality feature by tracking the number of occurrences and how software errors occur within a specified period.

2. Literature Review

A historical analysis and an applicative perspective on software reliability models can be found in bibliographic references of (Cheng, Bell & Liu, 1997) and (Lyu, 1996).

Since Jelinsky and Moranda have proposed the first reliability enhancement model (Jelinsky & Moranda, 1972), over the past 35 years, many SRGM models of reliability have emerged, among which are worth mentioning those in the exponential category, the Weibull and Gamma type,

those based on the infinite failure time, and last but not least the Bayesian models. (Lyu, 1996; Pham, 2010).

In practice, there is a tendency to establish a unitary methodology in the software reliability modeling process (Huang, Lyu & Kuo, 2013).

A major problem faced by these models is the achievement of a certain level of features such as ease of application and validation in contravention of the most accurate description of the technical procedures assumed to be used in real modeling processes. In order to reach software tools with a high level of reliability, two high-tolerance fault tolerance systems are pursued (Lyu, 1996).

Evaluating the failure probability within a specified timeframe is the primary focus of most of the models that are used in software reliability (Kaneva & Dimitrova, 2019). The model developed by Eckhardt and Lee (2005) has tracked how errors that occur in the design of various software systems have a beneficial influence on the possible relationship between errors, relative to various objectives imposed in the process of implementing them.

Determining the influence of errors in the various software design processes on the level of reliability is the main aspect of the model proposed by Littlewood & Miller, (1989). Dugan & Lyu, (2015) have proposed a rewarding Markovian model, which compares the reliability of the system obtained through various approaches to diversified design, and Tomek & Trivedi, (2005) have suggested a network-based squitter model with reward for error-tolerant software systems.

Popov, Strigini, and others (Popov, Strigini, May & Kuball, 2013) have estimated the upper and lower limits of the probability of failure in the diversified design based on the subdomain concept of the stress space. In the literature of (Cai, Lyu & Vouk, 2005; Cai & Lyu, 2004) there are comparative analyses and different assessments of the models used in the estimation of software reliability.

Each software uses a database. Thus, calculating the backup and restore efficiency (Kuyumdzhev, 2019) is important for software reliability.

3. Methodology of Software Reliability Engineering Modeling

Modeling the reliability of software systems over a prefixed timeframe is a major goal, and seeks to find an equation that adapts the data obtained by observing the errors that occurred during that timeframe.

Thus, it has been possible to use specific statistical methods in order to obtain the appropriate parameters for the models used in the reliability assessment and prediction process.

The main scenarios in the software reliability assessment models are:

- 1) The level of reliability must be established in the context of identical use and testing;
- 2) Ensure that the code is corrected and that direct errors are eliminated;
- 3) Provide the necessary framework to reduce the risk of new errors occurring when correcting the source code in order to eliminate the detected errors;

- 4) An equation will be sought to estimate the possible link between the errors in the source code and the probability of software failure caused by them.

Software Reliability Growth Models (SRGMs) are those models that are based on the assumption of reliability, with the increasing complexity and software testing time, which directly reduces the number of errors in software.

The practical application process of software reliability engineering principles is outlined in Figure no. 1.

The development and optimization of the framework in which users use software systems follows the quantitative estimation of reliability, correlated to the greatest satisfaction of user needs and requirements. Measuring the level of software reliability is the fundamental objective of the modeling process. In this respect, we are looking at how a software system approaches the estimated reliability level, when it is considered to be available to be implemented.

Starting from the data on the errors that cause the failures that were retained during the test phase, the reliability level of a software system is calculated by using several software reliability models.

Figure no. 1: Overview of the software reliability engineering process
(Source: adaptation after Lyu, 1996)

Within the software reliability engineering process, there are six major activities (Jelinsky & Moranda, 1972), presented in Figure no. 2.

At the bottom of the figure 1 and in parallel with the SRE process, the software development process is described, so that the activities of one are put in a relationship with those of the other process.

Both processes follow spiral patterns, but I will not present feedback paths for

simplicity. On the ground, some data is collected, which is used to improve the SRE process.

In Figure no. 1, are outlined four elements of greatest importance for software reliability engineering, namely:

- 1) the objective of reliability;
- 2) the operational profile;
- 3) modelling and measuring reliability and reliability;
- 4) validation of reliability.

(a) – Operational profile development (b) – Stages of Requirements and Architecture

Figure 2: *Overview of the software reliability engineering process*
(Source: Adaptation after Jelinsky & Moranda, 1972)

Activities 1-4 begin all at the stages of requirements and architecture development within the software development process. They extend to varying levels in the design and implementation phases. Activities 5 and 6 coincide with the test.

4. Applying the Capability and Maturity Model Integrated into Software Reliability Engineering

Capability Maturity Model Integration (CMMI) is a way of combining the features of system engineering, specific to multiple domains, and presents the five stages of development known as capability and maturity levels for development processes that are identified within an organization.

The CMMI model is applied within an organization and it monitors the progress of a project developed in a department of the organization or throughout the organization.

By using this model in the software reliability engineering model, we have pursued the development of the organization by optimizing the processes that take place within it.

Domains such as the acquisition and distribution of products, the multilateral and efficient development of processes and / or products and integrated systems engineering represent, in addition to software reliability engineering, the fundamental objective of process modernization efforts within

organizations specialized in the production of software.

The use of multiple Capability and Maturity Model variants (CMM) within the same organization, due to the notable differences between their operational profiles, generates major disadvantages due to the emergence of redundant information.

The CMMI model is included in the high complexity category, it has a structure based on a central component, to which new elements are added with the emergence of new areas where it can be applied.

The core component is associated with the field of system engineering and software engineering. This model has been implemented and implemented, starting from two distinct approaches. On the one hand, account was taken of the lack of flexibility of the model, due to the maturity levels associated with the implementation stages within organizations that seek to optimize processes according to the goals set for their own businesses.

On the other hand, it was intended that the process of implementing the new CMMI model, starting with the earlier version CMM v1.1, should be done at the lowest cost, which would not affect the investment efforts made by the organizations.

4.1. Objectives, Advantages and Limitations

The CMMI model has been implemented, starting from two distinct approaches.

In order to solve the problems mentioned above, in the process of application of the model, we used two distinct approaches, namely: the variant involving the application of the model in stages, namely the one that pursues the continuous application of the model. Both approaches of the model have similar content, in which we have found 22 Process Areas, obtained by developing the 18 CMM

v1.1 variant, pursue relatively the same generic and specific objectives, plus the implementation in the equivalent stage, offers the possibility of transposing the results obtained from the evaluation as necessary in the process of transition from the continuous application to the one phase.

Applying this model, a matrix representation, exposed in matrix M.1, of the relationships established between the process domains, relationships determined by the maturity levels present in the stepped application variant, respectively the domains of the process categories specific to the implementation variant continues.

Matrix M.1

The association between maturity levels and process domain categories (PACs)

The implementation variant continues

Step-by-step version of the CAP	PAC	Process Management	Project Management	Engineering	Technical support
	Maturity level				
	5 Optimizing	OID CL 0-5			CAR CL 0-5
	4 Quantitatively Managed	OPP CL 0-5	QPM CL 0-5		
	3 Defined	OPD CL 0-5 OT CL 0-5 OPF CL 0-5	RSKM CL 0-5 IPM CL 0-5	PI CL 0-5 RD CL 0-5 TS CL 0-5 VER CL 0-5 VAL CL 0-5	DAR CL 0-5
	2 Managed		PP CL 0-5 SAM CL 0-5 PMC CL 0-5	REQM CL 0-5	PPQA CL 0-5 CM CL 0-5 MA CL 0-5
	1 Initial				
	0 Incomplete				

4.2. Profiles of Capability Levels

In the continued application of reliability models, the capability profile is reached by individually assigning a capability level for each process area.

In practice, 22 capability levels are grouped into six categories, numbered from scratch (incomplete) to five (optimizable) categories that are associated with the respective domains of processes.

Matrix M.2 provides an example:

Matrix M.2.
Profile of capability levels

Step-by-step variant	Continuous variant									
	PAC	Process Management		Project Management		Engineering		Technical support		
	Maturity level									
	5	OID	CL 3					CAR	CL 1	
	4	OPP	CL 2	QPM	CL 0					
	3	OPD OT OPF	CL 1 CL 2 CL 2	RSKM IPM	CL 2 CL 1	PI RD TS VER VAL	CL 3 CL 4 CL 2 CL 2 CL 3	DAR	CL 3	
	2			PP SAM PMC	CL 2 CL 3 CL 3	REQM	CL 2	PPQA CM MA	CL 3 CL 3 CL 2	
	1									

4.3. Equivalent Equipping

There is a unidirectional association between the continuous and the staged

implementation. This is called equivalent staging.

Matrix M.3.
Profile of capability levels for maturity level 2

Step-by-step variant	Continuous variant									
	PAC	Process Management	The management projects		Engineering		Technical support			
	Maturity level									
	5	OID			CAR					
	4	OPP	QPM							
	3	OPD OT OPF	RSKM IPM	PI RD TS VER VAL	DAR					
	2		PP SAM PMC	CL 2 CL 2 CL 2	REQM CL 2	PPQA CM MA	CL 2 CL 2 CL 2			
	1									

It sets the standards for the capability levels to be achieved in order to be equivalent to a certain level of maturity in the continuous representation.

An example of the level of capability required for level 2 equivalent is presented by matrix M.3. Process Areas (PA) from

3-5 maturity levels are not of interest for assessing maturity level 2.

An example of the level of capability required for level 3 equivalent is described in Matrix M.4 (PA from maturity levels 4-5 are not of interest).

Matrix M.4.

Profile of capability levels for level 3 maturity equivalents

Continuous variant

Step-by-step variant	PAC	Process Management	The management projects	Engineering	Technical support
	Maturity level				
	5	OID			CAR
	4	OPP	QPM		
	3	OPD CL 3 OT CL 3 OPF CL 3	RSKM CL 3 IPM CL 3	PI CL 3 RD CL 3 TS CL 3 VER CL 3 VAL CL 3	DAR CL 3
	2		PP CL 3 SAM CL 3 PMC CL 3	REQM CL 3	PPQA CL 3 CM CL 3 MA CL 3
	1				

Due to the fact that an organization reaches a minimum of capability level 3 for PAs placed at that level and for those on the lower maturity level, the necessary framework for achieving the equivalence between level 4 or 5 maturity is created.

5. Conclusions

In the last period, many analytical models have been proposed and studied to evaluate the quality of software systems. Each model should make certain assumptions about the software development process as well as the test environment. The environment may change depending on the software application, the

development cycle, and the engineering design team's capabilities.

The existing technical methods of the SRE are suffering from several limitations, including:

- the use of a software product within an organization's own operating environment may lead to the collection of a small amount of data from the testing process that has nothing to do with errors that may lead to breakdowns. Software products with a high level of reliability require a complex test process deployed in a significant time span.

Estimation and reliability prediction due to the use of the limited set of test data may cause accuracy problems;

- the unrealistic evaluation of software reliability over its real value is determined by the incorrect assumptions underlying the methods currently used in the SRE domain.

Current models used in the software reliability estimation process are considered to be successful if they discover the ideal framework for their application.

Without validation on industrial domains, the exercise of modeling can only become of intellectual interest and cannot be adopted considerably in industry.

Thus, although SRE has been an engineering field for some time now, credible software reliability techniques are imperative, particularly for modern software systems.

REFERENCES

Bankov, B. (2019). Software Evaluation of PHP MVC Web Applications. *19 International Multidisciplinary Scientific Geoconference (SGEM): EXPO Proceedings, Vol. 19*, Albena, Bulgaria, 603-610.

Cai, X., & Lyu, M. R., (2004). An Empirical Study on Reliability Modeling for Diverse Software Systems. *IEEE International Symposium on Software Reliability Engineering ISSRE*, IEEE, 125-136.

Cai, X., Lyu, M. R., & Vouk, M. A. (2005). An Experimental Evaluation on Reliability Features of N-Version Programming. *IEEE International Symposium on Software Reliability Engineering ISSRE*, 161-170.

Cheng, J., Bell, D. A., & Liu, W. (1997). Learning belief networks from data: an information theory based approach. *Proceedings of the Sixth International Conference on Information and Knowledge Management*, Las Vegas, Nevada, USA, 325-331.

Dugan, J. B., & Lyu, M. R. (2015). Dependability Modeling for Fault-Tolerant Software and Systems. *Software Fault Tolerance, Vol. 15*, Wiley, 109-138.

Eckhardt, D. E., & Lee, L. D. (2005). A Theoretical Basis for the Analysis of Multiversion Software Subject to Coincident Errors. *IEEE Transactions on Software Engineering, Vol. 11, Issue 12*, 1511-1517.

Huang, C. Y., Lyu, M. R., & Kuo, S.-Y. (2013). A Unified Scheme of Some Nonhomogenous Poisson Process Models for Software Reliability Estimation. *IEEE Transactions on Software Engineering, Vol. 29, Issue 3*, 261-269.

Humphrey, W. S. (2009). *The Watts New? Collection: Columns by the SEI's Watts Humphrey*. Special Report CMU/SEI-2009-SR-024, available at: https://resources.sei.cmu.edu/asset_files/SpecialReport/2009_003_001_15035.pdf, accessed on 18 October 2019.

Jelinski, Z., & Moranda, P. B. (1972). Software Reliability Research, In Freiburger, W. (Eds), *Statistical Computer Performance valuation* (pp. 465-484). New York: Academic Press.

Kaneva, M., & Dimitrova, V. (2019). Telecommunications usage and energy consumption in Bulgaria. *19th SGEM International Multidisciplinary Scientific GeoConference, Vol. 5*, 741-748, <http://doi.org/10.5593/sgem2019/5.3>.

Kuyumdzhiyev, I. (2019). Comparing backup and restore efficiency in MySQL, MS SQL server and MongoDB. *19th International Multidisciplinary Scientific Geoconference SGEM: Conference Proceedings, Vol. 19*, 167-174.

Littlewood, B., & Miller, D. R. (1989). Conceptual modeling of coincident failures in multiversion software. *IEEE Transactions on Software Engineering, Vol. 15, Issue 12*, 1596-1614.

Lyu, M. R. (1996). *Handbook of Software Reliability Engineering*. New York: IEEE Computer Society Press and McGraw-Hill Book Company.

Lyu, M. R., & Cai, X. (2007). Fault-Tolerant Software. In Wah, B. W. (Eds), *Wiley Encyclopedia of Computer Science and Engineering*.

Pham, H. (2010). *Software Reliability*. Secaucus, New Jersey, USA: Springer-Verlag New York, Inc.

Popov, P., Strigini, L., May, J., & Kuball, S. (2013). Estimating bounds on the reliability of diverse systems. *IEEE Transactions on Software Engineering*, Vol. 29, Issue 4, 345-359, available at: <https://ieeexplore.ieee.org/document/1191798>, accessed on 25 September 2019.

Tomek, L. A., & Trivedi, K. S. (2005). Analyses Using Stochastic Reward Nets, In Lyu, M. R. (Eds), *Software Fault Tolerance* (pp. 139-165). New York: John Wiley & Sons Ltd, available at: <https://www.scribd.com/document/334107822/Software-Fault-Tolerance-Ch-6-pdf>, accessed on 10 October 2019.

Weill, L. (2008). Leverage and Corporate Performance: Does Institutional Environment Matter?. *Small Business Economics*, Vol. 30, Issue 3, 251-265.