2014 Fourth International Conference on Advances in Computing and Communications

Software development Life cycle model to improve maintainability of software applications

*Velmourougan S[a], Dhavachelvan P[b], Baskaran R[c] Ravikumar B [d]

*a & d – STQCIT, Chennai, STQC Directorate, Ministry of Information & Communication Technology, Govt. of India. India. E-mail: velmourougan@gmail.com

b- Department of Computer Science and Engineering, Pondicherry University, Puducherry, India. E-mail: dhavachelvan@gmail.com

c- Department of Computer Science and Engineering, Anna University, Chennai, India, E-mail: baskaran.ramachandran@gmail.com

***Abstract*— Software Maintainability is the business issue in large scale applications and it must be given adequate focus during software development process to minimize the downtime. Inculcating the maintainability features in the software application during its development can minimize the maintainability efforts during its real time use. In that view, this paper presents a new, Maintainable-Software Development Life Cycle model (M-SDLC) introducing maintainability development tasks or activities to be followed during the SDLC. This paper provides a set of activities and best practices for all stakeholders involved in the planning, architecting, coding and testing and maintenance software applications during development. This paper is also presented with a comparative study done on existing SDLC model and concludes that the present models are not adequately focusing on maintainability issues while building software products to minimize maintenance issues. We prove with statistical results that proposed new SDLC model is capable of building maintainable application.**

Keywords- Software Maintainability, SDLC, Software Design, Software maintenance criteria

I. INTRODUCTION

"The capability of the software product to be modified" is the definition for maintainability in ISO 9126 standard. The modification in the software application could be due to its requirement change or due to functional flaws during its use. Because maintainability demands for core changes in the existing systems to mitigate flaws it consume resources. If the software application is not built with the capabilities to handle maintainability changes demanded then, certainly it consumes more resources and makes the maintenance as a tedious task.

Software maintainability engineering must be given most important accent in the requirements definition, design, coding, testing and maintenance of systems development processes [5,6]. Software maintainability criteria demands for more complex and integrated development process that controls for its ease of solving issues at its live usage. If the main streams of controls are inculcated in development phases, as part of the maintainability engineering development process, including, Software Maintainability Requirements Analysis, Software Maintainability Design Analyses, Maintainable Coding, Software Maintainable Testing and maintenance.

To build an application with better maintainability features and paying more focus on designing the software maintainability to distinguish malfunctions, failures, flaws to inculcate easy mitigation strategies and to control operations/process dependent on applications [7-10]. We propose a new SDLC model. The model proposed in this paper lists the various tasks, actions and activities to be followed during the SDLC to build a sound maintainability feature.

A. Research questions addressed in this paper:

i. Whether existing SDLC models support to build maintainability features in the software product?

ii. Whether SLDC best practices improve the maintainability features into the software product?

iii. What are the key activities/best practices to be followed during SDLC to improve software maintenance?

In the rest of the paper section 2 provides "Research survey of existing SDLC Models", section 3 covers an "overview of the SDLC model development process", section 4 presents the "New M-SDLC Model" and concluded in section 5.

II. RESEARCH SURVEY OF EXISTING SDLC MODELS

As a part of research survey we have derived and listed a set of maintenance based development task as shown in Table 1. The development tasks are based on the engineering experience gained during the testing of large scale applications. The basis for the compilation of the development tasks is described in section 3 of this paper. On the other hand, we have chosen, few popular SDLC models, namely V-shaped, Common criteria, RUP, Scrum, Waterfall, Spiral, Prince2. [11-15] Based on the analysis over SDLC models a capability matrix as shown in Table 1 is derived. It is evident as shown in Table 1, that most of the above listed models do not come up directly with the maintainability features and they addressed very few development tasks as functionality issue. No specific focus is given to Maintainability factors. . Our research identifies that none of the above listed SDLC models have specific focus on safety factors.

Table 1 list the development tasks that are based on empirical Maintainability testing conducted in the process of evolving a new M- SDLC model. The Maintainability testing details are explained further in forthcoming sections.

978-1-4799-4363-0/14 $31.00 © 2014 IEEE
DOI 10.1109/ICACC.2014.71

The phases of SDLC must address various development tasks to build the product with maintainability capability. For example, in the Requirement Phase of SDLC, Determine Operational Profile, Maintainability assessment, Maintainability Modelling, Conduct Trade of studies, List Maintainability Modelling, Document Maintainability Requirement. The tasks are verified at the beginning of research survey to understand the capabilities of existing SDLC models and listed in Table 1 and it answers our first research question.

Table 1 Capability study of SDLC models based on Software Maintainability.

		SDLC Model								
Sl no	Development Tasks	V-shaped	on criteria	RUP	OWASP	Scrum	Waterfall	Spiral	Prince2	MS-SDL
1	**Requirement Phase**									
1.1	Determine Operational Profile	N	Y	N	N	N	N	N	N	Y
1.2	Maintainability assessment	N	Y	N	N	N	N	N	N	Y
1.3	Maintainability modeling	N	Y	N	Y	N	N	N	N	N
1.4	Conduct Trade of studies	N	Y	N	N	N	N	N	N	Y
1.5	List Maintainability Modeling	N	N	N	N	N	N	N	N	N
1.6	Document Maintainability Requirement.	N	N	N	N	N	N	N	N	N
2	**Design Phase**									
2.1	Maintainability modeling	N	Y	N	N	N	N	N	N	Y
2.2	Redundancy Analysis	N	Y	N	N	N	N	N	N	N
2.3	DRACAS	N	N	N	N	N	N	N	N	N
2.4	Design impact evaluation	N	Y	N	Y	N	N	N	N	Y
2.5	Maintainability Prediction	Y	Y	Y	Y	Y	Y	N	N	Y
2.6	Trade off analysis	Y	Y	Y	Y	Y	Y	N	N	Y
2.7	Document maintainability design criteria	N	Y	N	Y	N	N	N	N	Y
2.8	Document maintainability Checklist	N	Y	N	Y	N	N	N	N	Y
2.9	Verification of maintainability requirement	N	Y	N	Y	N	N	N	N	Y
3	**Coding Phase**									
3.1	Coding,	Y	Y	Y	N	Y	Y	Y	Y	Y
3.2	Static analysis measurement	Y	Y	Y	Y	Y	Y	Y	Y	Y
3.3	Unit testing &measurement	N	N	N	N	N	N	N	N	N
3.4	Verification of code for maintainability Requirement	Y	Y	Y	Y	Y	Y	N	Y	Y
4	**Testing Phase**									
4.1	Maintainability Evaluation	Y	Y	Y	Y	Y	Y	Y	Y	Y
4.2	Maintainability demonstration	Y	Y	Y	Y	Y	Y	N	N	Y
4.3	Track testing progress	N	Y	N	Y	N	N	N	N	Y
4.4	Estimation of MTTR	Y	Y	Y	Y	Y	Y	Y	Y	Y
4.5	Maintainability Verification Testing	Y	Y	Y	Y	Y	Y	Y	Y	Y
5	**Maintenance Phase**									
5.1	Document maintainability in user manual	N	N	Y	N	N	N	Y	N	N
5.2	Default configuration	N	N	Y	Y	N	N	N	N	Y
5.3	Customer liaison	N	Y	Y	Y	N	N	N	Y	Y
5.4	Fault logging	N	Y	N	Y	N	N	N	N	Y
5.5	List of updates	N	Y	N	Y	N	N	N	N	Y
5.6	Maintenance program review	N	Y	N	Y	N	N	N	N	Y

III. OVERVIEW OF THE SDLC MODEL DEVELOPMENT PROCESS

New Maintainability-SDLC (S-SDLC) Model is developed in three different phases, namely Initial, Validation (with the S-SDLC) and Reliability estimation phase as shown in Table 2. Our idea of verifying the proposed model is by comparing a software application-1 developed using a popular SDLC model {S1(W)**}** (Example: Software-1 developed using a waterfall model {S1(W)**}**) with {S1} (Software-1 developed using S-SDLC {S1}). Initially the application {S1 (W)**}** is tested for maintainability development tasks as shown in Table 1.

Table 2: SDLC Model development phases.

New SDLC Model development Phases	Description
Initial Phase	A product developed using a formal SDLC model is evaluated with the focus on maintainability test cases.
Validation (With S-SDLC Phase)	The product is subjected to complete development using the anomalies reported based on maintainability test results.
Reliability Estimation Phase	Final evaluation of the reliability of the application in terms of its ability to perform its intended function over a period of time. Estimation Mean Runs to Failure (MRTF).

A. Initial Phase

Three different fully built applications categorical namely standalone {S1(W)**}**, client-server {S2(W)**}** and the web based {S3(W)**}** are tested for the safety requirement test cases and recorded the failures in two categories, namely functionality failures Sx(f) and safety failures Sx(m) (where x denotes software identification number) as shown in Table 3. . The reported failures during maintainability testing are logged initially and the Root cause Analysis (RCA) on the failures are derived as Vulnerabilities as listed in Table 3. In the process of evolving new M- SDLC model the vulnerabilities are found during Maintainability testing of {S1 (W)}, {S2 (W)} and {S3 (W)}. The anomalies are uniquely identified using Vulnerability Identity (VID) values termed as VID (X) where X stands for the anomaly number. The Table 3 shows that out of 848 anomalies found in total with the combination of functionality and safety attributes as shown in Figure 2, only 171 anomalies are directly influencing to the failure of the application due to lack of maintainability controls. The vulnerabilities are listed in Table 3. Based on the anomalies listed and the mitigation methodologies are devised in the form of Key tasks and its associated activities for every phase of the SDLC as presented in S-SDLC Model in Figure 3.

1) Studies on vulnerabilities and results

The Table 3 shows the result of analysis carried out on the failures reported during the first cycle of the testing, assuming that the software application developed with the adoption of the waterfall SDLC model using both functional and non-functional cases. During the evaluation testing on the application developed using Waterfall SDLC model the anomalies are investigated and grouped to weed out the root causes of the failures. But the anomalies associated with functional and software maintenance issues are segregated and reported as shown in Table 3. The number of failures verses the security anomalies (vulnerabilities) are shown in Figure 2. The reported failures are depicting the two dimensional view on

anomalies, their priorities and their influences over the application found during its testing.

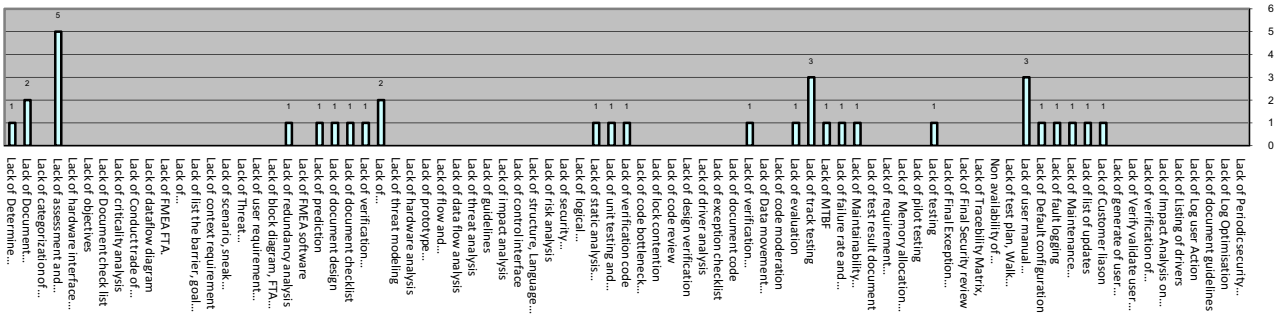

Fig. 2 The Number of security anomalies versus vulnerability with the Secure-SDLC Phase:

The applications with the similar functional requirements are developed again from the beginning by inculcating the M-SDLC model with the diligence of key tasks and its associated actions as indicated in Figure 3. The frozen application built using M-SDLC model is validated and verified for the maintainability test criteria and the reported failures are analysed to find suitable controls. Table 4 shows the statistics on improvement in the application adopting the proposed M-SDLC Model and the results in-turn answers our second research question.

Table 3: Statistics of vulnerabilities associated with the functionality and the safety.

Vulnerability description-Category	Functionality			Maintainability		
	S 1(f)	S 2(f)	S 3(f)	S 1(m)	S2(m)	S 3(m)
Lack of Determine operational profile	0	0	0	0	1	1
Lack of Document requirement	3	3	5	1	2	2
Lack of assessment and modeling,						5
Lack of redundancy analysis						1
Lack of prediction						1
Lack of document design			1			1
Lack of document checklist						1
Lack of verification requirements			1			1
Lack of modeling, DRACAS,						2
Lack of static analysis measure						1
Lack of unit testing and measure			1			1
Lack of verification code						1
Lack of verification requirement			2			1
Lack of evaluation			1			1
Lack of track testing			27			3
Lack of MTBF						1
Lack of failure rate and RGT						1
Lack of Maintainability, MRTF						1
Lack of testing			1			1
Lack of user manual document						3
Lack of Default configuration			1			1
Lack of fault logging						1
Lack of Maintenance program review						1
Lack of list of updates						1
Lack of Customer liaison						1

2) Applications used for development of SDLC Model

The aim of this study is to demonstrate that the application built using M-SDLC model is capable of doing away with the anomalies pertaining to Maintainability by inculcation the suitable control mechanism. {S1 (W)}, {S2 (W)} and {S3 (W)} were tested to qualify their Maintainability requirements for use. The authors took due care in selection of the applications for demonstrating the proposed M-SDLC model to fit into a wide range of application. Three categories of applications, namely standalone, client-server and the web based applications to determine their requirements for security. The coverage for the type of applications is mostly addressed with the above stated categories of applications. Embedded or platform specific application, namely mission critical military applications or simple gaming applications are covered under the standalone category, Financial, trading applications or time-stamped applications are covered under the category of client-server based application e-commerce and e-governance applications are covered under the category of web-based applications.

A squad of 11personnels involved in the development of the product and the complexity of the product ranges between 7000 to 12000 SLOC. In actual use around 6000 users are attracted by these applications. Forthcoming sections show that M-SDLC model provides best practices as a solution to the complete software development life cycle to not only coder, but also to requirement analysts, architects/architects, programmers, testers and maintenance engineers.

Table 4: Statistics on reduction of anomalies versus key functions of software (S1,S2 &S3).

Name of the Cell	No of Test Runs Estimated	MRTF (R)	Failure Status	
			Without the proposed Model	With proposed M-SDLC Model
Security Cell	38	750	84	4
File Based Cell	38	750	176	6
S/W with std.	38	750	76	1
Fixed Database	50	1000	153	1
Report/Analysis	77	1500	67	1
Error Handler	60	1200	121	3

Total			677	16

IV. NEW M-SDLC MODEL

A New M-SDLC (Maintainability -Software development life cycle) model addressing the key activities associated with every phase of the Software development life cycle to proactively defend the vulnerabilities as discussed in the above sections. The evaluation and SDLC developed in this paper is in line

with the waterfall model (for easy understanding) and the same can be further analysed to develop S-SDLC model based on other popular models like RAPID, Incremental etc. but the approach remains same. The Fig 3 provides the answer to our third research question that "What are the key activities/best practices to be followed during SDLC to improve software safety?"

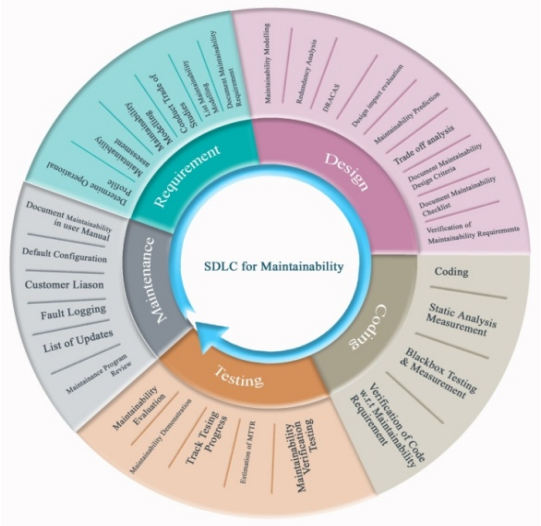

Fig. 3: M-SDLC model Proposed with Tasks/Activities

V. CONCLUSIONS

Maintainability in a software application is not only be improved by providing additional resources and increasing frequency of Preventive Maintenance but also needs the adoption of best practices during each phase of the software development life cycle. Software maintainability is a key quality attribute to improve the efficiency of timely evolvement of failures. The statistics shown in Table 4 is the clear evidences that the best practices have reduced the number of failures closer to zero in respect to maintainability failures. The study results in the form of statistics are evident by means of comparing the failure rate of the application with and without the best practices as shown in Table 4, that the failure rate per test run for the application has decreased effectively. The contribution of this paper is the compilation of best practices in the form of the M-SDLC model. The model enables the developer to track and mitigate the anomalies due to the lack of addressing Maintainability issues in the application during software development. The estimation of reliability using the reliability estimation model as reported in [16] with and without the best practices has proved that the M- SDLC Model can be adapted for application development to minimize or shield the downtime of the software application due to maintainability failures.

VI. REFERENCES

[1] Aggarwal, K.K. ; Singh, Y. ; Chhabra, J.K.,An integrated measure of software maintainability,Annual Reliability and Maintainability Symposium, Proceedings.Publication Year: 2002 , Page(s): 235 - 241

[2] Hegedus, P., Revealing the Effect of Coding Practices on Software Maintainability, 29th IEEE International Conference on Software Maintenance (ICSM), 2013,Publication Year: 2013 , Page(s): 578 - 581

[3] Liang Ping, A Quantitative Approach to Software Maintainability Prediction, International Forum on Information Technology and Applications (IFITA), 2010

[4] Volume: 1' Publication Year: 2010 , Page(s): 105 - 108.

[5] Sunday, D.A., Software maintainability-a new `ility',Annual Reliability and Maintainability Symposium, 1989. Proceedings., Publication Year: 1989 , Page(s): 50 - 51

[6] Zhuo, F. ; Lowther, B. ; Oman, P. ; Hagemeister,, J.Constructing and testing software maintainability assessment models, First InternationalSoftware Metrics Symposium, 1993. Proceedings.,Publication Year: 1993 , Page(s): 61 - 70

[7] Perepletchikov, M. ; Ryan, C. ; Frampton, K., Cohesion Metrics for Predicting Maintainability of Service-Oriented Software,Seventh International Conference on Quality Software, 2007. QSIC '07. Publication Year: 2007 , Page(s): 328 - 335

[8] Bowles, J.B., Code from requirements: new productivity tools improve the reliability and maintainability of software systems,Annual Symposium - RAMS Reliability and Maintainability, 2004,Publication Year: 2004 , Page(s): 68 - 72

[9] Davis A.M, Bersoff E.Hm Comer E.R, A strategy for comparing alternative software development life cycle models, IEEE Transactions on Software Engineering, Volume: 14 , Issue: 10 Publication Year: 1988 , Page(s): 1453 - 1461.

[10] Common Criteria Sponsoring Organizations, "Common Criteria for Information Technology Security Evaluation Part 1: Introduction and General Model, Version 3.1 Rev 1," Nat'l Inst. Of Standards and Technology CCMB-2006-09-001, Sept. 2006.

[11] Common Criteria Sponsoring Organizations, "Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Components, Version 3.1 Rev 1," Nat'l Inst. Of Standards and Technology CCMB-2006-09-002, Sept. 2006.

[12] Common Criteria Sponsoring Organizations, "Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Components, Version 3.1 Rev 1," Nat'l Inst. Standards and Technology CCMB-2006-09-003, Sept. 2006.

[13] N.R. Mead, E.D. Hough, and T.R. Stehney II, "Security Quality Requirements Engineering (SQUARE) Methodology," CMU/SEI, Technical Report CMU/SEI-2005-TR-009, ESC-TR-005-009, Nov. 2005.

[14] Branstad, Martha; Powell, Patricia B.; Software Engineering Project Standards, IEEE Transactions on Software Engineering, Volume: SE-10 , Issue: 1 publication Year: 1984 , Page (s): 73 – 78.

[15] IEEE Recommended Practice for Software Requirements Specifications, IEEE Computer Society, IEEE Std 830-1998.

[16] S Velmourougan, P Davachelvan, and R Baskaran, Software Reliability Qualification Model, International Journal of Performability. 8(2012) 437-446.