Maintainability is one of the most significant quality indicators. Its accurate measurement constantly support and increase the maintenance procedure. An accurate measure of software quality completely depends on maintainability measurement. A number of maintainability theories have been published till date and the maintainability concept has been developed with different research states. At this paper we will talk about some of the essential theories given by some researches in their paper and we will relate those all research with our paper in order to encourage our effort.

This paper shows the outcome of a systematic literature review conducted to accumulate the facts on software maintainability estimation of object oriented design. In this review paper, our endeavor is to discover the existing known comprehensive and complete model or framework for measuring the maintainability of object oriented design at an initial stage of development life cycle.

Keywords: Software Maintainability, Maintainability Estimation, Object Oriented Design, Software Quality, Software Maintenance.

[1] INTRODUCTION

Software maintainability has turn into one of the most significant concerns of the software industry. Maintainability is a key quality attribute of software systems [11]. On the other hand, maintainability has always been an elusive conception and its correct measurement or estimation is a complex exercise. Researchers and practitioners have always argued that maintainability should be considered as a key attribute in order to promise the quality software.

Maintenance and Maintainability are two dissimilar terminologies. Software maintenance is explained as “the procedure of modifying a software system or component after delivery to correct faults, advance performance or added attributes, or adapt to an altered environment. Whereas Software maintainability is defined as “the ease with which a software system or component can be modified to correct faults, advance performance or added attributes, or adapt to an altered environment [12].

This states that maintenance can be regarded as a route performed for the duration of SDLC process of a software, but maintainability is one of the key quality attribute [13].
Maintainability factor donate to 40-70% of the price of software products. Improved maintainability at all times guide to reduced maintenance efforts and reduced price and time.

[2] MAINTAINABILITY MEASUREMENT

On the other hand, maintainability has always been an elusive concept and its correct measurement or evaluation a complex exercise. The majority of the studies measure maintainability or precisely the attributes that have impact on maintainability at the source code level. Estimating maintainability at a later stage leads to the late arrival of desired information, leading to late decisions about changes in design. This basically increases cost and rework. Therefore, near the beginning, assessment of maintainability in the development process may improve quality and reduce maintaining efforts and costs. Practitioners repeatedly advocate that maintainability should be planned early in the design phase.

Maintainability estimation in the early hours at design phase is highly emphasized in my proposed study; hence, considered important for the delivery of quality software. Our main passion is that it is for the duration of the analysis and design phase that maintainability analysis can yield the highest payoff: design decisions can be made to improve maintainability earlier than implementation starts. When the design meets the maintainability requirements, it can be implemented and the constraints added for maintainability enhancement of the design and are required to be verified before maintaining.

There is a general agreement among industry professionals and academicians to join together maintainability with the development life cycle in order to deliver protected, safe and reliable software inside time and budget [5]. Our purpose is to present a comprehensive framework to help measuring and assessing maintainability in a practical manner, with a focus on the design stages of object-oriented development.

[3] CLOSELY RELATED WORKS

Broad range of maintainability calculation models have been planned in the literature within last two decades. A number of maintainability models/methodologies were proposed to facilitate the designers in measuring the maintainability of object oriented software so as to produce superior and improved software systems. Opening from 1970s to 2014 a variety of maintainability estimation models or techniques was developed.

In 1977, Jim McCall planned a software quality model called as McCall’s model. In this model McCall recognized the 11 quality factors broken down by the three key perspectives for characterizing the quality attributes of a software product namely product revision (ability to change), product transition (adaptability to new environments) and product operations (basic operational characteristics. For every quality factor McCall defined one or more quality criteria (a way of measurement), in this approach an overall quality estimation could be made of a given software product by evaluating the criteria for each factor. For instance the maintainability quality factor would have criteria of simplicity, conciseness, and modularity as the sub characteristics.

An additional software quality model called as Boehm’s quality model was given by Barry W. Boehm in 1978. He distinct a hierarchical model of software quality uniqueness, in
trying to qualitatively describe software quality like a set of attributes and metrics (measurements). At the uppermost level of his model, Boehm defined three most important uses namely, as-is utility, the extent to which the as-is software can be used (i.e. ease of use, reliability and efficiency), maintainability, easiness of identifying what needs to be changed as well as ease of modification and retesting and portability, effortlessness of changing software to provide somewhere to stay a new environment.


Oman et.al (1992) Coleman et.al (1994) confirmed that how software maintainability study can be used to direct software related decision making. Li.W et.al (1993) used impression of regression to compute maintainability of software systems. Welker K. et.al (1997) completed that MI should not be interpreted in a vacuum slightly it should be used as an indicator to direct human investigation. Muthanna et al. (2000) developed a maintainability model via polynomial linear regressions. But this model could be useful only for procedural software and not appropriate for object-oriented software.

Polo et al. (2001) used number of modification needs, mean effort per modification appeal and type of correction to check maintainability. M. Dagpinar et.al (2003) accomplished that size and import direct coupling metrics are important predictors for measuring maintainability of classes although inheritance, cohesion and indirect/export coupling measures are not. Di Lucca et.al (2004) provided WAMM (Web Application Maintainability Model) exact to web applications only.

On or after the survey of literature it has been observed that a variety of researchers proposed numerous models for maintainability evaluation, but in nearly all of these works, maintainability evaluation based on the measures taken after the coding phase of development life cycle. For the reason that of this, maintainability predictions are ready in the latter stages of development life cycle, and it turn out to be very difficult, hard and costly to improve the maintainability at that stage.

A Critical Look of Maintainability Models Consider by Various Expert

<table>
<thead>
<tr>
<th>Study/Author</th>
<th>Year</th>
<th>Maintainability Approach</th>
<th>Measurement</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muthanna Model fully based on Polynomial Linear Regression</td>
<td>2000</td>
<td>A number of Design Level Metrics</td>
<td>Not best suited to maintainability estimation at design phase</td>
<td></td>
</tr>
<tr>
<td>Huffman Hayes representation(Observable-Mine-Adopt (OMA))</td>
<td>2003</td>
<td>Based on Maintainability product,Perceived maintainability</td>
<td>Which result has no any understandable relationship with</td>
<td></td>
</tr>
<tr>
<td>Model Description</td>
<td>Methodology</td>
<td>Year</td>
<td>Comments</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Lucca- Fasolino WAMM(Web Application Maintainability Model)</td>
<td>Apply Size metrics, Control coupling metrics, data coupling metrics, Complexity metrics</td>
<td>2004</td>
<td>Method was very difficult and has high Complexity.</td>
<td></td>
</tr>
<tr>
<td>Hayes- Zaho Model(MainPredMo)</td>
<td>Through MI(Maintainability Index), LOC(Lines of Code), TCR(True Comment Ratio), AC(Attribute Complexity), CR(Comment Ratio)</td>
<td>2005</td>
<td>Unfortunately all above maintainability metrics measure only higher level of abstraction.</td>
<td></td>
</tr>
<tr>
<td>Koten-Gray Model(Bayesian Network Maintainability Prediction Model )</td>
<td>Focus on Line Of Code, SIZE1, Lack of Cohesion of Methods, Data Abstraction Coupling, Depth of Inheritance, Weighted Method per Class, Message Passing Couple, etc</td>
<td>2006</td>
<td>Maintainability Prediction Model apply at later stage of SDLC</td>
<td></td>
</tr>
<tr>
<td>Zhou -Leung Model MARS (Multiple Adaptive Regression Splines)</td>
<td>Compared with other Support vector models, Multivariate linear regression models, Regression tree models and Artificial neural network models and proved to be best.</td>
<td>2007</td>
<td>But this model has various limitations from implementation point of view.</td>
<td></td>
</tr>
<tr>
<td>Prasanth Ganesh Dalton Model</td>
<td>With the help of FRT(Fuzzy Repertory Table) followed by Regression analysis</td>
<td>2008</td>
<td>This hypothesis was not empirically validated.</td>
<td></td>
</tr>
<tr>
<td>MO. Elish and KO Elish</td>
<td>Produced Treenet model using stochastic gradient boosting</td>
<td>2009</td>
<td>Sorry to say, these achievements have not been widely accepted.</td>
<td></td>
</tr>
<tr>
<td>C Jin, JA Liu</td>
<td>Based on Support vector machine</td>
<td>2010</td>
<td>Wasteful of resources, and requires unduly long times.</td>
<td></td>
</tr>
<tr>
<td>Gautama Kang Model (COMPOUND MEMOOD MODEL)</td>
<td>During Line Of Code, Difference Line of Code, Maintainability Index, Cyclomatic Complexity</td>
<td>2011</td>
<td>This is practically very expensive</td>
<td></td>
</tr>
<tr>
<td>Alisara Hincheeranan, Wanchai Riverpiboon</td>
<td>Used design size, design hierarchy, abstraction and encapsulation, coupling, cohesion etc, through Maintainability Estimation Tool (MET),</td>
<td>2012</td>
<td>But this model is not validated for better acceptance.</td>
<td></td>
</tr>
</tbody>
</table>
Monika Saini, Mukti Chauhan 2013 Targeted various parameters including both source code metrics and design metrics Presented the evidence of various maintainability techniques or models

[4] CRITICAL OBSERVATIONS

Following above winning completion of the literature review a number of important explanations can be enumerated as follows.

1. If we estimate the software maintainability at an early stage that is design phase in the software development life cycle, significantly improve the software quality and as well as client happiness, and decrease overall cost, time and effort of rework.
2. In order to reducing effort in measuring maintainability of object oriented design we require to recognize a minimal set of maintainability factors for object oriented development system, which have optimistic impact on maintainability measurement.
3. Object oriented software characteristics are required to be recognized and after that the set of maintainability factors appropriate at the design phase should be finalized.
4. Further, maintainability metrics have to be chosen at the design phase for the reason that metric selection is an important step in maintainability estimation of objects oriented design.

[5] CONCLUSION

A number of approaches have been proposed in the literature for measuring software maintainability. An investigation of the considerable literature shows that greatest efforts have been place at the later stage of software development life cycle. A resolution to modify the design in order to improve maintainability after coding has started is very costly and error-prone. Therefore, it is an understandable fact that estimating maintainability early on the development procedure may significantly reduce maintenance cost, time, effort, and rework. The early estimation of maintainability at design phase can yield the highest payoffs. On the other hand, the lack of maintainability at early stage may not be compensated during subsequent development life cycle.

After the above conversation our conclusion is that maintainability is a quality factor that attempts to calculate how much effort will be required for software maintaining and to estimate the difficulty of causing a fault to result in a failure. The goal of increasing the maintainability of software is not just to detect defects but more significantly, to identify defects as soon as they are introduced. As a consequence, sinking the cost and time to fix the bug and producing higher quality maintainable software each build of the release. After an exhaustive review process we found that reducing effort in measuring maintainability of object oriented design is must in order to deliver quality software within time and budget.
[6] CONTRIBUTION

The most important contribution of this chapter is in the field of maintainability measurement. We have conducted a systematic review in this field. The dissimilar factors maintainability and measurement for these factors are identified. Overall contribution is listed as follows:

Systematic Literature Review

A complete step by step improvement of the systematic review procedure is described. It will help to further researchers as a reference for undertaking SLR.

A. Recognition of key papers related to the maintainability study in software engineering domain
B. Discovery of maintainability factors and measurement in the recent domain of OOD
C. Identification and arrangement of different concepts about the software maintainability in the present software engineering domain.
D. A proposed software maintainability framework to assist the self-assessment for designers to identify software maintainability factors.
E. Structure and well defined assessment process for finding factors from high level to lower measurable level.
REFERENCES


N. Soni and M. Khaliq
