Volume 5, Issue 3, March 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Survey on Selection of Metrics to Measure Changeability

Preetika Gupta*, **Rohit Chahal**
CSE Department, HEC Jagadhri
Kurukshetra University, Haryana, India

***Abstract--** The software maintenance is a major cost concern. The maintainability of a system seems to have much impact on the ease or difficulty to implement changes. A consensus has emerged that the maintainability of a software system is dependent on its design in the procedural paradigm as well as in the object-oriented (OO) paradigm. Maintainability has four components which are stability, analyzability , testability, and changeability. In this our objective is to study and performance evaluation of different approaches to measure the changeability by using different metrics.*

***Keywords—**Object-Oriented(OO), Metrics, Changeability, Maintainability, CBO, Change Proneness.*

I. INTRODUCTION

Changeability means the ability of an operation system to alter autonomously the configuration to meet new, previously unknown demands e. g. from the market. Changeability is then the ability to realize new states of the in, out and throughput. Additionally, the system‘s reconfiguration has to be realized as quickly as the environmental changes.Hence, to be changeable, the speed of alteration is significant. Software applications are not single components systems, they are made up of collection of components. These components should be made in such a way that the code written once can be reused many times by doing some changes in the code , which can reduce the development time ,cost and effort. Sometimes changes introduce a new fault that degrades the functionality of the system. If a large change increment is done, it will introduce many new faults that will restrict the useful change delivered in the new version of the system.[4].

II. REASONS FOR CHANGES

Software evolution, adaptive, and corrective maintenance is common reason for changes. Often such changes cluster around key components. It is hence important to analyse the frequency of changes to individual classes, but importantly, to also identify and show related changes in multiple classes. During software development series of changes are made to software. Changes can be done due to a variety of reasons such as enhancements, adaptation, and perfective maintenance. Some parts of the software may be more prone to changes than others. Knowing which classes are changes prone can be very helpful; change-proneness may indicate specific underlying quality issues. If a maintenance process can identify what parts of the software are change-prone then specific remedial actions can be taken. The maintainability of a system seems to have much influence on the ease or difficulty to implement changes. A consensus has emerged that the maintainability of a software system is dependent on its design in the procedural paradigm as well as in the object-oriented (OO) paradigm. Maintainability has four components which are stability, analysability, testability, and changeability. In application areas like software systems and telecommunications are developing constantly. The assessment of the changeability of software systems is of major concern for buyers of large systems found in fast moving domains[5]. One way of assessing changeability is to assess the impact of changes earlier; systems were evolved by using structured approach, which was very successful, but just for simple applications. Then came the object-oriented (OO) approach, which is based upon polymorphism, encapsulation and inheritance. The use of object-oriented (OO) technology for evolving software has become quite widespread. Researchers assert that OO practice assures good quality software, that is, particularly software that is easy to reuse and extend.[4]

III. METRICS TO MEASURE CHANGEABILITY

Changeability is the ease with which a source code can be modified. It is evaluated through metrics calculated from the history of changes made. These metrics tell how well or bad is the change for the project in terms of it degree however, the selection of these metrics is matter of real concern as they must be chosen in such a manner that they must measure the true image and state of the software components with respect to the basic principles of software stability with openness for further change components of software are like organic compounds that change internally and externally with multiple environmental and business reasons, for usage of software to continue, and for it to remain non obsolete it needs to remain constant state of change to remain in sync with the real business life. The utmost concern is maintenance and further development of the software without conflicts, issues and bugs , therefore , when software components undergo adaptation , enrichment and feature additions there is always a risk of too much change leading to change is overall structure and form of the software itself that it may lead to the huge burnout between the stake holders of the project in progress [3]. Changeability refers to how easy it is to perform a specified modification. Change-prone classes in software require more observation because they require more effort, development and maintenance costs. Identifying such classes can enable developers to focus protective actions such as testing and restructuring efforts on the classes that are sensitive and change prone. As a result, developers can deliver higher quality products in a timely manner by efficiently utilizing the resources. Modification may be on account of diverse factors like improvement, modification, perfect upkeep or do away with drawbacks. Several elements of the software may be susceptible to modifications than their counterparts. Managing change is one of the pivotal factors in the realm of software engineering. [6]

A. VARIOUS METRIS USED TO MEASURE CHANGEABILITY

Definition of various metrics are shown in table below:

Table I Definitions of various metrics

S.NO.	**METRIC NAME**	**METRIC DEFINITION**
1.	Coupling Between Objects (CBO)	CBO for a class is a count of the number of other classes to which it is coupled and vice versa..
2.	CA (Afferent Couplings**)**	CA is a measure of how many other classes use the specific class.
3.	CE (Efferent Couplings**)**	CE is a measure of how many other classes are used by the specific class.
4.	Weighted Methods Per Class(WMC)	The WMC is a count of the sum of complexities of all methods in a class.
5.	Changeability Quality Measurement (CQM*)*	CQM is a measurement of change applies to a class, to a variable or to a method.
6.	Extensibility Quality Measurement (EQM*)*	EQM is systemic measure of the ability to extend a software design principle where the implementation takes into consideration future growth.
7.	Depth of Inheritance (DIT)	The depth of a class within the inheritance hierarchy is the maximum number of steps from the class node to the root of the tree and is measured by the number of ancestor classes.
8.	Number of Children (NOC)	The NOC is the number of immediate subclasses of a class in a hierarchy.
9.	Response For a Class (RFC)	RFC is a count of methods implemented within a class and the number of methods accessible to an object class due to inheritance
10.	Lack of Cohesion of methods(LC OM)	For each data field in a class, the percentage of the methods in the class using that data field; the percentages are averaged the subtracted from 100%.
11.	Number of Public Methods (NPM)	NPM is number of local (not inherited) public methods.
12.	Message Passing Coupling(MPC)	MPC is a number of messages sent by a class in direction of the other classes of the system.
13.	CBO Using(CBOU)	CBOU refers to the classes used by the target class.
14.	CBO Is Used By(CBOIUB)	CBOIUB refers to the classes using the target classes
15.	CBO No Ancestors(CBONA)	CBONA is CBO without considering the classes ancestors
16.	Ancestors Method–Method Import Coupling(AMMIC)	AMMIC is number of parents classes with which a class has an interaction of the method-method type and a coupling of the type IC.
17.	Others Method–Method Import Coupling(OMMIC)	OMMIC is number of classes (others that super classes and subclasses) with which a class has an interaction of the methodmethod type and a coupling of the type IC.
18.	Descendants Method–Method Export Coupling(DMMEC)	DMMEC is number of subclasses with which a class has an interaction of the method-method type and a coupling of the type EC.
19.	Others Method–Method Export Coupling(OMMEC)	OMMEC is number of classes (others that super classes and subclasses) with which a class has an interaction of the methodmethod type and a coupling of the type EC.

20.	Value Weighted Filtered Outdegree (VWFO)	VWFO identifies the valuably flexible designs by combining changeability with the value as operationalized by utility changes.
22.	Number of Attributes per Class(NOA)	NOA is the total number of attributes/variables defined in the class.
23.	Number of Instance Method (NIM)	NIM is the total number of Instance Methods.
24.	Number of Instance Variable (NIV)	NIV measures the relations of a class with other objects of the program.
25.	Number of Local Methods (NLM).	NLM is number of local (not inherited) methods
26.	Number of Local Default Visibility Methods (NLDM)	NLDM is number of local default visibility methods.
27.	Number of Private methods (NPRM)	NPRM is number of local (not inherited) private methods.
28.	Number of Protected Methods (NPROM)	NPROM is number of local protected methods.
29.	Number Of Lines (NL)	NL is number of all lines.
30.	Blank Lines Of Code (BLOC)	BLOC is number of blank lines of code.
31.	Source Lines Of Code (SLOC)	SLOC is the number of lines that contain source code.
32.	Lines of Declarative Code (LDC)	LDC is number of lines containing declarative source code.
33.	Lines of Executable Code (LEC)	LEC is number of lines containing executable source code.
34.	Lines of Comment (LC)	LC is number of lines containing comment.
35.	Statement Count (SC)	SC is total number of declarative and executable statements.

IV. LITERATURE REVIEW

A. Metrics As Good Predictors Of Changeability.

YirsawAyalewet.al[1] attempted to investigate the usefulness of three coupling metrics (CBO, Ce, and Ca) and one size/complexitymetric (WMC) as predictors of changeability. The evaluation was based on a case study of open source softwareknown as OpenBravoPOS which is commonly used in the retail business and developed in Java. The results showed that some of the coupling metrics can be used as good predictors of changeability. **Franck Xia et.al**[8] articulated the theoretic difficulties with the existing metrics designed for predicting software maintainability. To overcome the difficulties, they proposed to measure a purely internal and objective attribute of code, namely change impact dependency,and show how it can be modeled to predict real change impact.

B. Measurement Of Change Proneness.

AnkitaUrvashi et. Al[3] had found non-linear data fitting algorithm bi-square robust that gives best possible results as it more accurate with realistic data with trend and pass through most of statistical test of significances, thus they get a promising approach to measure change-proneness of the software development process ehiich has not been used previously. **DeepaGodaraet. Al[**6] aimed to provide a basis to improve the process of prediction of change prone classes..Their paper provided an extensive review of studies related to change proneness of software. The main goal and contribution of the review was to support the research on prediction of change prone classes. In addition, they provided software practitioners with useful estimation guidelines (for e.g. classes predicted to be more change prone require more effort).

C. Change Impact Analysis.

Bixin Li et.al [9] presented a comparative framework including seven properties, that characterized the CIA techniques, and identified key applications of CIA techniques in software maintenance.Their need for further research was also presented in the specified areas: evaluating the existing CIA techniques and proposing new CIA techniques under the proposed framework and developing more mature tools to support CIA, comparing current CIA techniques factually with unified metrics and common benchmarks, and applying the CIA more substantially and effectively in the software maintenance phase.**Hassan Osman Ali et.al** [12] investigated issues of change impact analysis process and identifies and compares current practice issues of a software change impact analysis, by assessing their strengths and weaknesses. They also reviewed existing tools and models of change impact analysis and how it supports to the current software change managements. Hence, they proposed existing process issues and suggested the need of an effective process of software change management.

D. Different Approaches Of Changeability.

KhineZar Ne Winn[2] aimed to measure the changeability and extensibility ofaspect-oriented (AO) software implemented in AspectJ. The analysisis based on MobileMedia, AspectJ projects, by using Self-OrganizingMap (SOM).**AnuradhaPanjetaet.al**[4] defined software as asset of programmes which contains set of instructions to form software. Software maintenance is that part of software which maintenance every part of it and reduce problems. Maintainability has four components, namely, analysability, testability, stability, and changeability.They identify the quantum of indexes which belongs to particular project.**AnuradhaPanjetaet.al**[5] defined changeability as a measure of impact of changes made to a module on the rest of the system.Changeability is one of the charactersitics of maintainability.They are solving this issue by building a framework which helps to measure degree of changeability by using clustering methods(machine learning). **K. Abdiet. Al**[7] proposed a probabilistic approach using Bayesian networks to answer to the problematic of change impact analysis and prediction in Object-Oriented systems. The built probabilistic model was tested on data extracted from a real system. To verify their approach, they took a correlation hypothesis between coupling and change impact already verified in previous works. **Dr. Ruchika Malhotra et.al** [10] attempted to reuse the generated prediction model of one project and validate it on another project. For the purpose of their evaluation, they had used two open source projects written in Java language. The performance of the predicted models was evaluated using Receiver Operating Characteristic (ROC) analysis .**Matthew E. Fitzgerald et al** [11] presented a new approach for exploring system changeability using Epoch-Era Analysis, with a prominence on tradespace exploration using an illustrative case study. Epoch-Era Analysis is an approach for explaining systems over time as existing in a series of static contexts (epochs) that change stochastically. A five-step method was used to produce an intuitive and accessible set of data and graphs explaining the abstract concept of changeability, which is hard to grasp due to its intrinsically time-dependent nature. Ideas for future changeability metrics were discussed.

Table II Evaluation Of Various Changeability Measurement Techniques

AUTHOR	YEAR	METRICS	SIGNIFICANCE	IMPLEMENTED ON	METHOD
M.K. Abdi et al.	2009	RFC,MPC,CBOU,CBOIUB,CBO,CBONA,AMMIC, OMMIC, DMMEC,OMMEC	To predict and analyze change impact in object-oriented systems	Data Extracted From a Real System	Bayesian Network
Matthew E. Fitzgerald et al.	2011	VWFO	To identify valuably flexible systems in tradespace studies in order to improve decision making during the conceptual design phase.	XTOS	Epoch Era Analysis
Dr.Ruchika Malhotra et al.	2013	CBO.NOC,NOM,NOA,NIV,NIM,NLM,RFC,NLDM, NPRM,NPROM,NPM,NL, BLOC,SLOC,LDC,LEC,LC,SC,DIT,LCOM,WMC	To reuse the generated prediction model of one project and validate it on another project	Frinika and Freemind	ROC Analysis
Yirsaw Ayalew et al.	2013	CBO,CE, CA,WMC	To access modularity of system.	Open Bravo POS	No Particular Method
Khine Zar Ne Winn	2014	CQM, EQM	To measure the changeability and extensibility.	Aspect Oriented Software	Self Organizing Map
Ankita Urvashi et al.	2014	WMC,DIT,NOC,CBO,RFC,LOCM,CA,NAPM	To optimize the datasets .	Data Sets	Bisqaure Method

V. CONCLUSION

Changeability is defined as a measure of impact of changes made to a module on the rest of the system and its evaluation is done through metrics calculated from the history of changes made. These metrics tell how good or bad is the change for the project in terms of it degree however, the selection of these metrics is matter of real concern as they must be chosen in such a manner that they must measure the true image and state of the software components. This issue can be solved by building a framework by selecting appropriate metrics for a given type of project which helps to measure the degree of changeability.

REFERENCES

[1] Yirsaw Ayalew&KagisoMguni, *An Assessment of Changeability of Open Source Software*, Computer and Information Science; Vol. 6, No. 3; 2013.

[2] KhineZar Ne Winn, *Quantifying and Validation of Changeability and Extensibility for Aspect-Oriented Software,* International Conference on Advances in Engineering and Technology (ICAET'2014) March 29-30, 2014 Singapore.

[3] Ankita Urvashi , Anamika Chhabra, *Predicting Changeability in Software Components Using Bisquare Method for Optimization*, International Journal of Advanced Research in Computer Science and Software Engineering Volume 4, Issue 9, September 2014.

[4] AnuradhaPanjeta,Prof.AjayKumar, *Performance Analysis to Changeability of Measuring Software Components,* International Journal Of Engineering Research & Management Technology July- 2014 Volume 1, Issue-4.

[5] AnuradhaPanjeta,Prof.AjayKumar, *Comparative Performance to Changeability of Measuring Software Components*, International Journal of Advanced Research in Computer Science and Software Engineering Volume 4, Issue 8, August 2014 .

[6] DeepaGodara , R.K. Singh, *A Review of Studies on Change Proneness Prediction in Object Oriented Software*, International Journal of Computer Applications (0975 – 8887) Volume 105 – No. 3, November 2014 .

[7] M.K. Abdi, H. Lounis, H. Sahraoui, *A Probabilistic Approach for Change Impact Prediction in Object-Oriented Systems,* AIAI-2009 Workshops Proceedings .

[8] Franck Xia, Praveen Srikanth , *A Change Impact Dependency Measure for Predicting the Maintainability of Source Code,*Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04) 2004 IEEE.

[9] Bixin Li , Xiaobing Sun, Hareton Leung and Sai Zhang, *A survey of code-based change impact analysis techniques*, 2012 John Wiley & Sons, Ltd.

[10] Dr. Ruchika Malhotra, Megha Khanna, *Inter Project Validation for Change Proneness Prediction using Object Oriented Metrics,* An International Journal (SEIJ), Vol. 3, No. 1, april 2013.

[11] Matthew E. Fitzgerald, Adam M. Ross, Donna H. Rhodes, *A Method Using Epoch-Era Analysis to Identify Valuable Changeability in System Design,* 9th Conference on Systems Engineering Research Los Angeles, CA, April 2011.

[12] Hassan Osman Ali, Mohd Zaidi Abd Rozan, Adamu Abubakar, Akram M. Zeki,Abdullahi Mohamud Sharif, Mueen Uddin and Jamshed Memon, *Assessing Issues of Change Impact Analysis Process for a Software Projects,* World Applied Sciences Journal 28 (10): 1366-1374, 2013 ISSN 1818-4952 © IDOSI Publications, 2013.