Volume 4, Issue 8, August 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Comparative Performance to Changeability of Measuring Software Components

Anuradha Panjeta*
CSE& Kurukshetra University
Haryana, India

Prof. Ajay Kumar
CSE& Kurukshetra University
Haryana, India

Abstract— Changeability is defined as a measure of impact of changes made to a module on the rest of the system. Changeability is one of the characteristics of maintainability. Factors include coupling between the modules, lack of code comments, naming of functions and variables. In this research work we are solving this issue by building a framework which helps to measure degree of changeability by using clustering methods (machine learning) , which would be best suited for our problem definition and better from previous research works.

Keywords*— Object-oriented (OO), The *within-cluster sum of squares (WCSS*), *Service-Oriented Computing (SOC).

I. INTRODUCTION

Software evolution, adaptive, and corrective maintenance is common reason for changes. Often such changes cluster around key components. It is therefore important to analyse the frequency of changes to individual classes, but importantly, to also identify and show related changes in multiple classes. During software evolution series of changes are made to software. Changes can be 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, namely, analysability, testability, stability, and changeability. In application areas like telecommunications, software systems are evolving constantly. The assessment of the changeability of software systems is of major concern for buyers of large systems found in fast moving domains. Designing and maintaining systems in a dynamic contemporary environment requires a rethinking of how systems provide value to stakeholder's time over. Developing either changeable or classically robust systems are approaches to promoting value sustainment. But ambiguity in defined across system domains has resulted in an inability to specify, design, and verify to utilities that promote value sustainment. Changeability, in contrast, 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 throughput and in-, out-.The reconfiguration of the system has to be realized as quickly as the environmental changes. Therefore, to be changeable, the speed of adaptation is important Software applications are not only single components systems, they are made up by collection of components. These components should be developed 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 new faults that degraders the functionality of the system. If a change increment is proposed, it will introduce many new faults that will limit the useful change delivered in the new version of the system. The rate of change of the system is therefore governed by organization decision making process.

II. BASIC PROPERTY OF CHANGE

The basic properties of change that are recorded by a simplest change management system include ownership (who made change) object that is changed (product, subsystems, module, file, set of lines) and time when the change was performed.[1]A change event can be characterized with three elements: (1) the agent of change, (2) the mechanism of change, and (3) the effect of change. The agent of change is force the instigator, for the change. The role of change agent can be implied or intentional but always requires the ability to set a changes in motion. The mechanism of change describe the path taken in order to reach state 2 from including any costs, state 1,both time and incurred ,money. The effect of change is the actual difference between the origin and destination states. This modification can be fine grained (a handful links of code in few files done by one developer of the task) or more coursed grained like the two differences between two releases of the system, spaced by several months.

III. TO MEASURE IMPACT OF CHANGE

Since change is hard to track and measure its impact on the software process needs a framework that works to get us influence on all aspects of software design, where and how it is influencing, what will be the implications, consequence of its influence. By collecting data before and after the change you implement, evaluate, you can measure and compare your agency's progress with respect to the goals you set out. The process of measuring changes should speed the improvement process; you should begin with simple measures rather than spending time developing a complex measurement system. Software change impact analysis has gained considerable attentions with recent challenges of the situation. As the software community recognizes the growing need to identify consequences of these changes impact analysis is making it way into the software process. Dependencies between software life cycle objects are becoming more numerous and complex as many software systems grow beyond a million of code of lines. Software change efforts are plagued with widely varying estimates for implementing software changes, since the impacts of the change are not readily in advance. The introduction of software change impact analysis into the software process adds more fidelity to software change visibility enabling more accurate software change estimates. Here we define where impact analysis is applied during software changes and describes how impact analysis can be addressed in the software process.

A software project gets affected due to change: Complexity and the need for conformity can make changing software an extremely difficult tasks. Changing one part of a software system often results in undesired side effects in other parts of the system, requiring more changes before the software can operate at maximum efficiency Also, a designer typically will need to make tradeoffs between two or more characteristics when designing the system. Consider highly modularized code m8stmrthis code is usually easy to maintain has a good changeability characteristic, but may not perform as well. On a similar vein a normalized database may not perform as well as a not normalized database. These trade off need to be identified so that informed design decisions can be made.

IV. MACHINE LEARNING AND SOFTWARE ENGINEERING

Machine learning, a branch of artificial intelligence are about the construction and study of systems that can learns from data. In machine learning, unsupervised learning refers to the problem of trying to find hidden structures in unlabelling data. Since the example given to there is no error, the learner are unlabeled, reward signal to evaluate a potential solution. This distinguishes unsupervised learning from reinforcement learning and supervised learning. Unsupervised learning is closely related to the problem of density estimation in statistics. The unsupervised learning also encompasses many other techniques that seek to summarize and explain key features of the data. Many method employed in unsupervised learning are based on data mining methods used to per-process data. Given a set of observations x1, x2,... …, xn, where each observation is a d-dimensional real vector and k-means clustering aims to partition the n observations into k sets ($k \leq n$) S = {S1, S2, …, Sk} so as to minimize the within-cluster sum of squares (WCSS):wiki

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$$

where μi is the mean of points in Si.

Machine learning can be used in software engineering projects, machine learning algorithms can be used to develop assessment model for projects .Machine learning helps in building programs that need to be upgraded with some changes through experience. Machine learning and changeability are related in the sense that some changes are done in software components machine learning is used so that program can adapt to the changes done in the program. Machine learning deals with the issue of how to build computer programs that improve their performance at some tasks through experiences. Machine learning algorithms have proven to be of great practical value in a variety of application domains. Not surprisingly the field of software engineering turns out to be a fertile ground where many software development and maintenance tasks could be formulated as learning problems and approached in terms of learning algorithms.

V. RELATED WORK

Shu-Ching Chen; Gulati, S.; Hamid, S.; Huang, X.; Luo, L.; Morisseau-Leroy, N.; Powell, M.D.; Chengjun Zhan; Chengcui Zhang, "A three-tier system architecture design and development for hurricane occurrence simulation," Information Technology: Research and Education, 2003. As an environmental phenomenon, hurricanes cause significant property damage and loss of life in coastal areas almost every year. Research concerning hurricanes and their aftermath is gaining more and more attention. The potential changeability of hurricane data and hurricane models requires robust, maintainable and easily extensible software system for hurricane simulation. Focusing on the design and implementation of the components in each layer, we describe a hurricane simulation system built on the three-tier architecture to achieve good flexibility, extensibility and resistance to potential changes.

Paulisch, F., "Software architecture and reuse-an inherent conflict?," Software Reuse: Advances in Software Reusability, 1994. Proceedings., Third International Conference on , vol., no., pp.214,, 1-4 Nov 1994. Software architecture focuses on the overall structure of a software system. The structure is defined by its components and the relationships between them and is often described in shorthand by such terms as model-view-controller, client-server, pipe and filter, etc. We focus not only on the reusability components themselves, but on the reusability of their interconnections as well. The interconnections could take on various forms e.g. aggregation, inheritance, frameworks, etc. Software architecture goes beyond the scope of classical software design in which the structure of a software system is mainly defined by a functional decomposition of its primary subject matter. Software architecture explicitly considers not

only the functional but also the non-functional aspects of software systems, such as reusability, changeability, maintainability, portability, interoperability, testability, efficiency, fault-tolerance etc. Desirable non-functional properties have a strong influence on the architecture of the software system. Careful consideration should be given to the relative importance of the multiple non-functional requirements at an early stage in the software development lifecycle because their objectives are often contradictory. We focus on how the non-functional property reusability relates to the software architecture of a system

Fitzgerald, M.E.; Ross, A.M., "Sustaining lifecycle value: Valuable changeability analysis with era simulation," *Systems Conference (SysCon), 2012 IEEE International,* vol., no., pp.1,7, 19-22 March 2012
This paper details a method involving a series of metrics and visuals designed to assist in the determination of a single design or set of designs' valuable changeability in the era domain of Epoch-Era Analysis (EEA). A brief introduction to the necessary concepts of EEA is included, with references for further information. Examples are provided in the form of a partial case study of a potential orbit-realigning space tug system. The method is shown to effectively distinguish the lifecycle value of designs of interest, quantify the value and frequency of use for the various change mechanisms, and offer a value for the tradeoffs between initial costs and lifetime returns associated with including change mechanisms. Reference [8] is a companion paper (recommended to be read first), detailing additional metrics and methods used to value system changeability in the multi-epoch domain. Multi-epoch analysis is best suited for understanding the performance of systems across the space of potential future uncertainties when considering their ability to change design; era analysis uncovers additional time-dependent information related to lifetime value and applied change mechanism usage.

Perepletchikov, M.; Ryan, C., "A Controlled Experiment for Evaluating the Impact of Coupling on the Maintainability of Service-Oriented Software," *Software Engineering, IEEE Transactions on* , vol.37, no.4, pp.449,465, July-Aug. 2011.One of the goals of Service-Oriented Computing (SOC) is to improve software maintainability as businesses become more agile, and thus underlying processes and rules change more frequently. This paper presents a controlled experiment examining the relationship between coupling in service-oriented designs, as measured using a recently proposed suite of SOC-specific coupling metrics and software maintainability in terms of the specific sub characteristics of analysability, changeability, and stability. The results indicate a statistically significant causal relationship between the investigated coupling metrics and the maintainability of service-oriented software. As such, the investigated metrics can facilitate coupling related design decisions with the aim of producing more maintainable service-oriented software products.

Poshyvanyk, D.; Marcus, A., "The Conceptual Coupling Metrics for Object-Oriented Systems," *Software Maintenance, 2006. ICSM '06. 22nd IEEE International Conference on* , vol., no., pp.469,478, 24-27 Sept. 2006.
Coupling in software has been linked with maintainability and existing metrics are used as predictors of external software quality attributes such as fault-proneness, impact analysis, ripple effects of changes, changeability, etc. Many coupling measures for object-oriented (OO) software have been proposed, each of them capturing specific dimensions of coupling. This paper presents a new set of coupling measures for OO systems - named conceptual coupling, based on the semantic information obtained from the source code, encoded in identifiers and comments. A case study on open source software systems is performed to compare the new measures with existing structural coupling measures. The case study shows that the conceptual coupling captures new dimensions of coupling, which are not captured by existing coupling measures; hence it can be used to complement the existing metrics.

Geppert, B.; Mockus, A.; Robler, F., "Refactoring for changeability: a way to go?," *Software Metrics, 2005. 11th IEEE International Symposium* , vol., no., pp.10 pp.,13, 1-1 Sept. 2005
Legacy systems are difficult and expensive to maintain due to size, complexity, and age of their code base. Business needs require continuously adding new features and maintaining older releases. This and the ever present worry about feature breakage are often the reason why the sweeping changes for reversing design degradation are considered too costly, risky and difficult to implement. We study a refactoring carried out on a part of a large legacy business communication product where protocol logic in the registration domain was restructured. We pose a number of hypotheses about the strategies and effects of the refactoring effort on aspects of changeability and measure the outcomes. The results of this case study show a significant decrease in customer reported defects and in effort needed to make changes.

VI. PROBLEM FORMULATION AND PROJECT ANALYSIS

When a car is manufactured, thousands of components are developed to get the final complete car. Most of the components of the cars are developed in such a way that these components are interchangeable and can be used in any another car if needed and if one component is not functioning properly it can be interchanged with other component by plugging in and out with other components, so that changeability of the components is increased and maintenance can be made easier , It has been long cherished that software development can also be done like this but it is really difficult to achieve this and more challenging even to measure the changeability factors that can help to achieve a manufacturing model based on the car manufacturing model, Therefore, it is imperative to organize and work on this aspect of software development especially when multiple stake holders are involved and conflicts are inevitable . There is an urgent need to understand how these software components can be implemented as plug and play devices and changeability of software components can be understand in some tangible framework. Therefore, in this research work we are solving this issue by building a framework which helps to measure degree of changeability by using clustering methods (machine learning) , which would be best suited for our problem definition and better from previous research works.

During analysis it is found that 2 projects were having observations less than 100 so we removed them on the basis of graph analysis,for providing better analysis and accurate result it is important to give equal oppurtunity to all the projects.

i) SCRIBE OAUTH LIBRARY

DATE	ADDITION	DELETION
10/10	2.5K	250
11/10	475	90
12/10	260	150
01/11	350	50
04/11	300	75
07/11	50	10
10/11	190	5
01/12	600	600
04/12	1.4K	1.35K

ii)VEREKIA-INITIALIZER

DATE	ADDITION	DELETION
04/11	14K	-
07/11	10K	12K
10/11	-	-
01/12	18K	18K
02/12	50K	3K
04/12	21K	19K

Now we choose only those projects which need change proneness. After observations of these data we select 4 projects for identifying the impact of changeability.

VII. TOOLS AND EXPERIMENTAL SETUP USED

(1) MATLAB-MATLAB "verson" is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and applications. The language, tools, and built-in math functions enable you to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages,
such as C/C++ or Java®. You can use MATLAB for a range of applications, including signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology. More than a million engineers and scientists in industry and academia use MATLAB, the language of technical computing.

Key Features: High-level language for numerical computation, visualization, and application development Interactive environment for iterative exploration, design, and problem solving Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations Built-in graphics for visualizing data and tools for creating custom plots Development tools for improving code quality and maintainability and maximizing performance Tools for building applications with custom graphical interfaces**.**Functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET, and Microsoft®Excel**.**

(2) STATISTICS TOOLBOX - Statistics Toolbox provides algorithms and tools for organizing, analyzing, and modeling data. You can use regression or classification for predictive modeling, generate random numbers for Monte Carlo simulations, use statistical plots for exploratory data analysis, and perform hypothesis tests.For analyzing multidimensional data, Statistics Toolbox includes algorithms that let you identify key variables that impact your model with sequential feature selection, transform your data with principal component analysis, apply regularization and shrinkage, or use partial least-squares regression.
Statistics Toolbox includes specialized data types for organizing and accessing heterogeneous data. Dataset arrays store numeric data, text, and metadata in a single data container. Built-in methods enable you to merge datasets using a common key (join), calculate summary statistics on grouped data, and convert between tall and wide data representations. Categorical arrays provide a memory-efficient data container for storing information drawn from a finite, discrete set of categories.

(3) **Netbeans 7.1.3: Key Features**

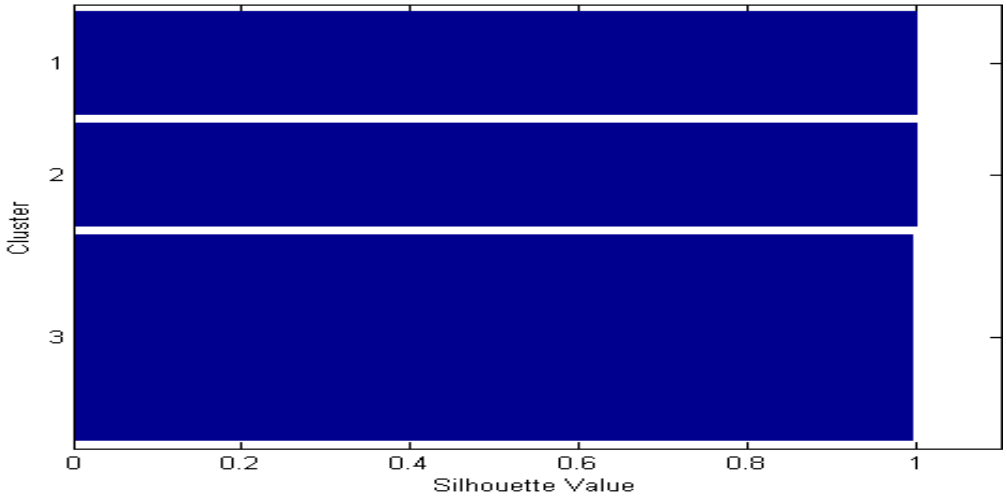
- Statistical arrays for storing heterogeneous and categorical data.
- Classification algorithms, including boosted and bagged decision trees, k-Nearest Neighbor, and linear discriminant analysis
- Probability distributions, including copulas and Gaussian mixtures
- Random number generation
- Hypothesis testing
- Design of experiments and statistical process control

Hardware Requirements:

- Intel Pentium p6 100 or higher
- Memory: 512 MB or above
- Hard disk drive: 10 GB or above
- Color monitor for visual display
- Mouse
- Keyboard

VII. RESULTS

Changeability of the software as show in graph based on silhouette values:



VIII. CONCLUSIONS AND FUTURE WORK

It is apparent from the graph that the actual values labeled for each sample of observation has been fairly learned by the k-means by identifying its centroids of clusters, then by looking at the index assignments after clustering, we see that class c which represents low possibility of change proneness has the highest no. of indices. Secondly, in this research work we have tried to identify the quantum of indexes which belongs to particular project, we find that the project named "jedis1" has the largest no of indexes included in the class of high proneness to charge where as project which have smaller n nature are less prone to instability and therefore less prone to change. Thirdly,, it also found that it does not matter how long the project have been worked on, the instability and change to proneness may creping at any point of time, as to is apparent from the statistics of addition and deletion from the github site. Fourthly, we found that the large no of indexes of each sampled observation belong to how proneness cluster, from which we can infer that the major part of the most of the are less prone to change but some part of (module, package)may prone to instability and proneness to change. All this may be attributed to the internal conflicts between the various contributors to this project.

In future, the research work may be considered for also calculating the risk associated with proneness to change using metrics which can calculate severity, occurrence or the frequency of the possible changes and timely detection for corrective measure which would lead to regain of stability and maturity of the software development. We can also implement model FMECA (failure mode efforts and criticality analysis), models for identifying the relationship between change proneness of the software to the total estimated risk.

REFERENCES

[1] Ingram, C.; Riddle, S., "Using early stage project data to predict change-proneness," *Emerging Trends in Software Metrics (WETSoM), 2012 3rd International Workshop on* , vol., no., pp.42,48, 3-3 June 2012

[2] Sebastian G. Elbaum John C. Munson,; "Code Churn: A Measure for Estimating the Impact of Code Change".

[3] The Challenges Of Software Change Management In Today's Siloed IT Organizations, A Commissioned Study Conducted By Forrester Consulting On Behalf Of Serena Software, November 2006

[4] N. Fenton, S. L. Pfleeger and R. Glass, "Science and Substance, A Challenge to Software Engineers", IEEE Software, July 1994, pp.86-95.

[5] D. German, A. Hindle, and N. Jordan. Visualizing the evolution of software using softChange. In *Proc. of the 16th Internation Conference on Software Engineering and Knowledge Engineering (SEKE 2004)*, pages 336–341, 2004.

[6] Nary Subramanian, Lawrence Chung;, "Metrics for Software Adaptability" 2Department of Computer Science University of Texas, Dallas Richardson, TX, USA,

[7] Gentzane Aldekoa1, Salvador Trujillo2, Goiuria Sagardui1, Oscar Díaz2"Experience measuring maintainability in software product lines"; XV Jornadas de Ingeniería del Software y B.ases de Datos JISBD 2006

[8] Chidamber, S. R. and Kemerer, C. K. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, Vol. 20 (June 1994), pp.476-493.

[9] Ajrnal Chaumun, M.; Kabaili, H.; Keller, R.K.; Lustman, F., "A change impact model for changeability assessment in object-oriented software systems," *Software Maintenance and Reengineering, 1999. Proceedings of the Third European Conference on* , vol., no., pp.130,138, 1999

[10] J. C. Munson, S. G. Elbaum, and R. M. Karcich, "Software Risk Assessment Through Software Measurement and Modeling", will appear *Proceedings of the 1998 IEEE Aerospace Applications Conference*, IEEE Computer Society Press.

[11] en.wikipedia.org/wiki/Cluster_analysis

[12] Jukka Kainulainen "Clustering Algorithms: Basics and Visualization"HELSINKI UNIVERSITY OF TECHNOLOGY Laboratory of Computer and Information Science T-61.195 Special Assignment 1

[13] Jon Kleinberg;" A Theorem for Clustering";, Department of Computer Science Cornell University

[15] MA Chaumun, RK Keller, F Lustman - Advances in software engineering, 2002 - dl.acm.org

[16] Bieman, J.M.; Andrews, A.A.; Yang, H.J., "Understanding change-proneness in OO software through visualization," Program Comprehension, 2003. 11th IEEE International Workshop on , vol., no., pp.44,53, 10-11 May 2003

[17] Kabaili, H.; Keller, R.K.; Lustman, F., "Cohesion as changeability indicator in object-oriented systems," Software Maintenance and Reengineering, 2001. Fifth European Conference on , vol., no., pp.39,46, 2001

[18] Adam M. Ross, Donna H. Rhodes, Daniel E. Hastings, Volume 11, Issue 3, pages 246–262, Autumn (Fall) 2008

[19] 2007Changeability in operations: A critical strategic resource for European manufacturing " Universities Engineering Conference, 2007. UPEC 2007. 42nd International , vol., no., pp.930,937, 4-6 Sept

