Software Design Changes: Analysis and Impact of Design Changes on Software Design Performance and Quality

Muhammad Abdullah Awais
MS150200157, Virtual University of Pakistan
ms150200157@vu.edu.pk

Abstract— **Development of quality software is not merely coding some pages according to the requirement and delivering to the client Rather quality software development is a complete process which leads towards finished product from requirement to maintenance. Many software fail despite of conformance to requirements because their design has not any support for changes made in design and architecture due to some changes in requirements or any other reason in final phases of development. Quality software design architecture is prepared by following quality matrices and frameworks to adapt the changes made at any phase of development because change in software development is something that cannot be avoided despite of careful requirement engineering and analysis. Change in design occurs due to change in any process leading to software design and most of the times change comes from client in requirement. This change leads to change in design of software under development and impact overall software design and its architecture to adjust the change and many a times this subsequent change in design leads to bad software design. This bad software design leads to poor quality software. Sometimes design changes come in bulk and it becomes difficult to track those changes made by different designers and developers which leads to difficult situation when changes are needed to be tracked. So to overcome these issues, in this research paper we will review the existing literature to define the impact of changes in requirement and design on the design of software. We will propose a system to track these changes that will be reflected in design and software performance metric will be studied in relation to these changes leading to bad software design because if change and impact of that change in design is not carefully examined, will lead to bad software design that will be costly to rectify and eventually would brunt the design and schedule to deliver the product.**

Keywords—**Software Design, Software Process, Performance Metric, Quality Metric, Software Design Methods, Change Management, Bad Design and Requirement Tracking**

I. INTRODUCTION

Software industry is growing at rapid pace and with its growth its opening new doors for the business to achieve their goals efficiently. Requirement gathering and software design process is becoming more and more efficient. Yet with this advancement there are some crucial activities in software development that are unavoidable at any case and one of them is the change in requirement and changes made in software design due to changes in requirements.

Design of software lays the foundation of any software and it determines how flexible and maintainable software is. Because maintainability, adaptability and flexibility of software determines the life time of software because if software cannot adapt to changes, it will not last longer in the industry and it will conclude the bad design of underlying software. That's why quality design promotes quality software. Design must be of high quality and should be open for changes but sometimes changes comes in bulk and it becomes difficult to cope up with these changes which results in bad design of software.

To overcome these issues, different design performance metrics are utilized to access the quality of software design during all the stages of development. Requirement and design changes are always inevitable at any stage of software development so its developer's precautions and innovation to design the software in such a way that it adapt according to the changes and that's why design metrics are employed to help comprehend design problems and to make predictive models of that design in efficient manners. Through the start phase of design development, changes are always expected so to design the better software, first phase of requirement engineering is very important because it's the phase where most of the design problems are rooted and if it is done efficiently, there is less chance to creep them in design phase in happy scenario. In this research paper, our focus would be on the impact of design changes due to requirement changes on design of software, reasons of design changes and efficient ways to track these design changes. We would try to explore the design issues and study metrics to measure the performance of software design. We will study a simple framework to track design changes.

This research paper is divided in different sections. In section I introduction is given with essential detail while section II introduce with related work done on this topic and some background knowledge which leads our research review in right direction so that any information technology personnel not having background knowledge should not have any difficulty in interpreting this research review. In

section III research problem will be stated for which whole research has been conducted. Section III will have research methodology and research question. In section III, I have defined the hybrid research model for conducting this research which involved review planning, research needs, selection of articles, data extraction and results. In section IV a unique approach to requirement change management will be discussed that is helpful in managing requirements leading to change in design in development process. Alongside this approach we will discuss software quality attributes that must be satisfied for quality design and the modal of requirement engineering would be set to meet these quality attributes by satisfying each attribute. In section V we will discuss a simple approach to track design changes and section VI will describe quality metric suit. In section VII conclusion and discussion will be presented.

II. RELATED WORK AND BACKGROUND

Purpose of this section is to introduce basic knowledge about knowledge management, software process improvement and enterprise resource planning to lead our research review also related work done on these topics is also referenced here to depict an overall picture.

As we are going to explore the impact of changes in design due to requirements and effect of these changes in first phase, that's why we need to review literature and related work done on these fields. For any software development project, design depends on the requirements and requirement engineering process comprises of several activities to be followed such as requirement elicitations, negotiation and analysis, specification document, validation and requirement document as discussed in detail in [1]. Author has provided a detailed analysis of prioritization of requirements in [1] because un prioritized requirements are one of those requirements that might find their way in making the design of software unstable.

Whenever a change is made in requirement in design phase, its leads to change in design and may cause bugs in software design and it happens due to the gap left between software design and requirements design which cause the system design to be faulty and ambiguous [2].To diminish effects made by these changes there must be a sound design strategy for crucial design assessment [2]. Due to these reasons, developers and designers face a lot challenges in requirement engineering process of requirement prioritization and classification to develop analysis model because non functional requirements directly relate to the design of software as these requirements put certain constraints on the design of software being developed [2].

To overcome these requirements issues, Nedhal Al-Saiyd and Esraa Zriqat has proposed a technique to managing the changing requirements in relation to the design of software [3]. Shahid and Muhammad Khalid in [4] has explained a quality metric suit to measure and analyse the performance of software design and will be exploring their findings to find the solution to problems of bad software design due to the inevitable changes introduced in middle of software development process.

III. RESEARCH METHODOLOGY

To answer the question on which my research is based, I performed extensive literature review according to the research guidance provided by B. Kitchenham [5]. Many research papers are presenting comparative studies while in some papers, new techniques are proposed. Approach used in this research paper is a hybrid version of systematic literature review and systematic mapping study as elaborated in fig 2. This hybrid model enabled me to review the latest research in the field of software design, software design process improvement, impact of changes on software design quality and performance analysis if software design while undergoing change management while exploring the knowledge from previous research in a systematic manner.Instead of pure analysis of software design techniques and performance metric comparatively; main focus was on understanding the impact of changes in design of software in final phases which might lead to critical issues in the form of bad software design with continuous process improvement and its implementation. So to focus on the result an overview and essential detail of some new and already used techniques is presented in this paper to answer the research question Whole process of research is described in following section.

A) Review Planning

Planning is an essence and surety of any research task. For a systematic review and systematic study planning as defined by B Kitchenham [5], I followed the research review planning protocols confirming to a successful review planning. A huge research has been conducted on the topic so to review the research we needed to carefully plan our research. Planned review included the steps of research identification by identifying the research review needs, selecting the quality research articles from top research article publishing platforms and indexing websites then extraction and synthesis of selected data governing the protocols of review planning.

B) Research Identification

Despite of the huge research available on the selected topic of impact of design changes on software design, it was necessary to find the links and success factors behind the use of management of software design leading to successful design. Tracking the design change being an important factor of software process improvement and design, contribute in the success of quality software product. It was needed to organize the research available in software design and design change management to define and elaborate the importance of quality of software design leading to the success of any system such. So I tried to review the research and identified potential benefits of software design quality and performance metrics. Review of the research leads to research question followed.

RQ: what are the impacts of design changes in software design in design phase due to changes in requirements and how to diagnose bad software design due to design changes?

C) Selection Criteria

A lot of research is available on the topic of software design and design improvement and software design has

undergone huge research and large numbers of research papers are available on these topics but it's required to select research based on our selection criteria that meet our research needs. In first phase I researched different online databases for the required research articles which include both the research article indexing service provider as well as the most famous research publishing forums. Fig 1 shows basic model followed for the research paper selection. I researched articles from notable libraries such as Higher education commission digital library, ACM digital library, Springer and on online indexing databases such as Google scholar and research gate using keywords as mentioned in keywords section. I used keywords and queries such as (software design changes, change management in software design, bad software design, software performance metric) and their alternative of these keywords were also considered along with "AND","OR" query clauses to get maximum results. This way numbers of research papers were 96.

To include in our review and research I applied different measures for inclusion and exclusion of research papers. Only those article were selected which conformed the research topic, abstract and our research intent. It was necessary to exclude those research papers which did not conform to our research intent and not have empirical research.

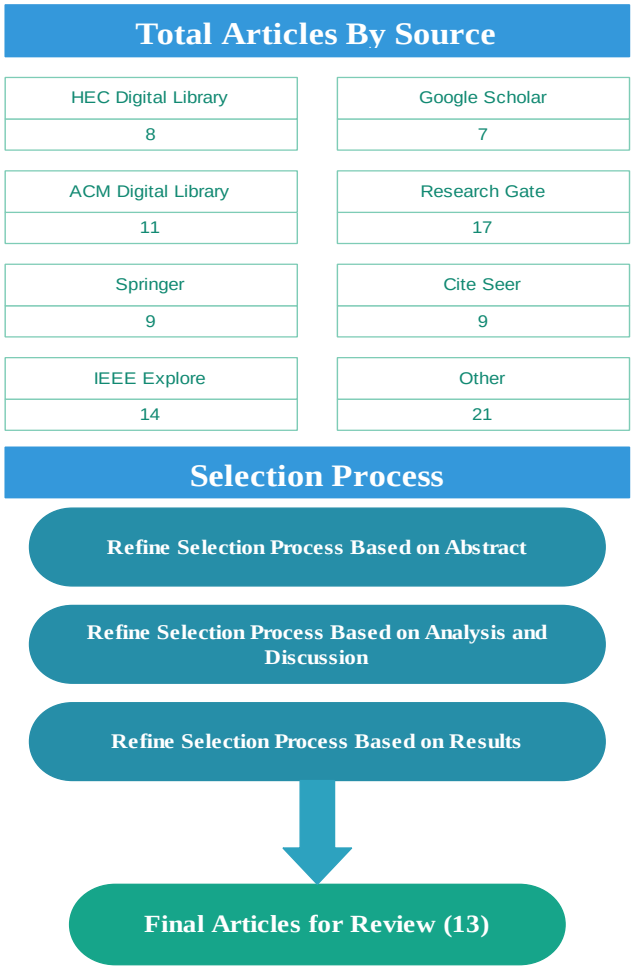

Fig. 2. Articles Selection process for Research

D) Data Extraction

In data Extraction step it was necessary to extract data related to the topic from all the selected research papers confirming the topic. Related data was extracted according to the relevance to terms in such a way that final data contain unique research result from the selected research papers.

E) Synthesis

After data extraction, it was necessary to synthesize the data by evaluating the resulted data against research identification and research question. Data was evaluated against the terms, research direction and main problem which further summarize the bulk data for further review. Without synthesis, it was not possible to review the results from previous research because huge research is available on this topic and it's difficult to summarize such enormous quantity of data. For synthesis of results, I used methods and guidelines such as textual narrative synthesis, thematic synthesis, Meta studies and Meta narrative explained in detail by theory with practice of tools and techniques for arranging the research for evaluation and synthesis in [6].

F) Review Results

After the extensive literature review by following the hybrid research model, analysis and exploration of different factors affecting the design of software showed requirements as one of the factors introducing changes in software design leading to some flaws and defects in design and its require corrective actions in the form of requirement tracking and management using framework where requirements are traced and managed into design in efficient manners. Further it's required to study software quality metrics to measure the performance and correcting bad software design.

If requirement gathering phase is followed efficiently then there are less chance of requirements leading to unnecessary design changes causing bad software design issues and if requirements are unavoidable then we must find a way as discussed to track the changes in design of software and take corrective actions to remedy the design by measuring and comparing the performance and quality of implemented design with the help of quality software matrices like ck metric suite.

Our research analysis showed that there are quality attributes regarding the software design that must be met for quality design so in upcoming sections we will discuss all the possible and efficient solutions to these problems and answer our research question. Detail analysis of quality attributes is presented in section IV where we summarized the quality attributes and discussed requirement management technique that has been found in literature and it's helpful in managing requirements. In later section, design tracking solution is presented as well as CK Metric suit is discussed that is used to find bad design anomalies which must be corrected for quality design.

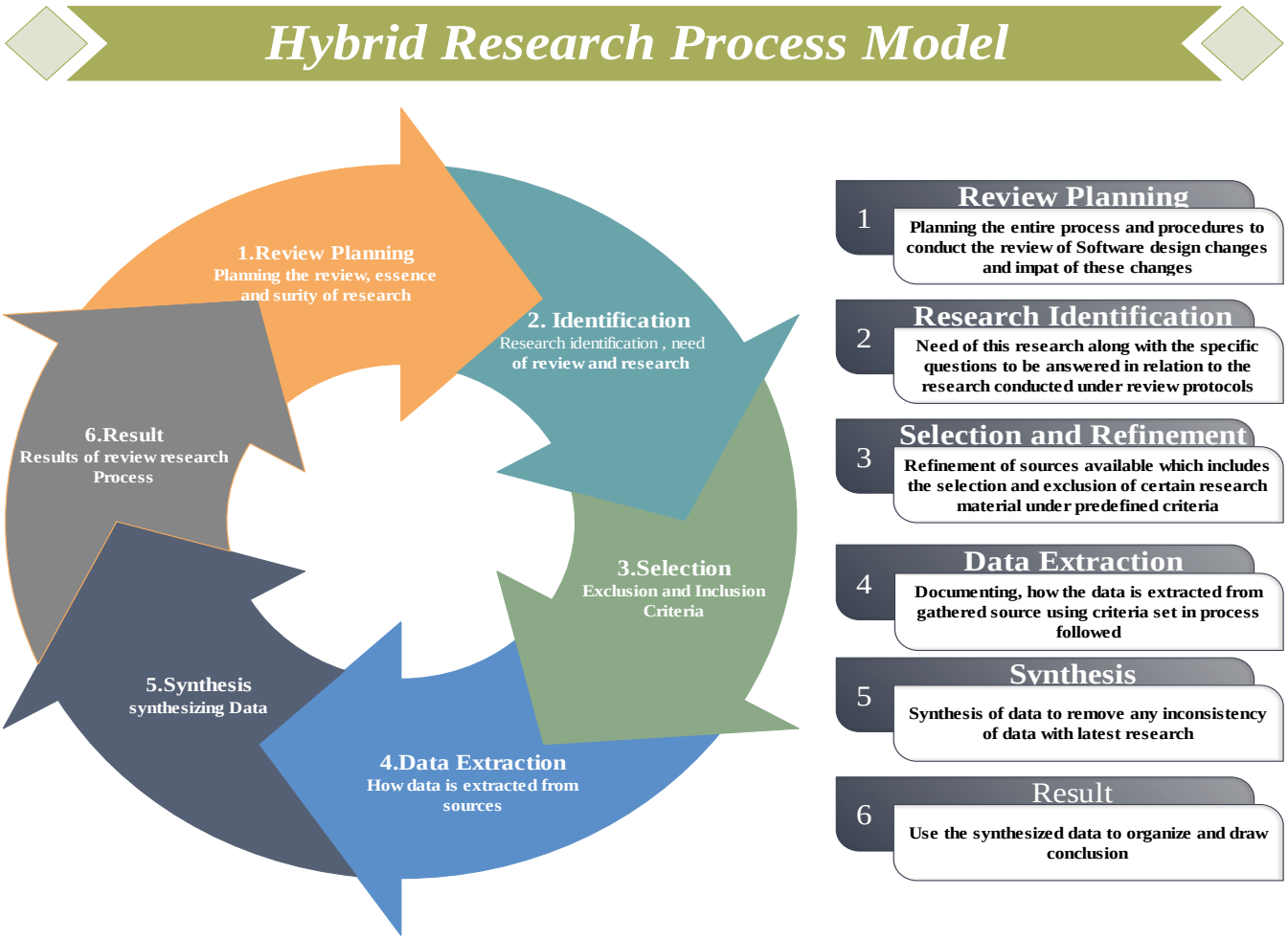

Fig. 2. Basic Model for Research

IV. QUALITY DESIGN ANALYSIS

After Fig. 3 shows software design quality attributes that must be depicted by a quality software design. It's difficult meat these qualities attributes due to the high frequency of requirement changes in design phase yet these attributes lay the foundation of quality software design. When requirement changes at design level, design becomes unstable and to make it more stable it must be modifiable in a way to reflect changes in requirements and it can only be modifiable if it exhibits maintainability and flexibility. In conclusion these attributes are linked to each other. One attribute follow another. So changes in requirements and other development processes such as schedule, methodology, budget, environmental constraints directly affect the quality attributes of software design and if not handled properly, it will lead to faulty design scenario. For example if schedule is shortened then there must be some quality assurance activities that will be sacrificed and in result design would be unproductive and usability of design will be lost. From very start of software development, these attributes must be under serious consideration at every phase of development whether its requirement engineering or design modal.

Fig. 3. Software Design Attributes

Otherwise system design would be unstable and extensibility would be compromised which is unacceptable. A detailed specification of these quality attributes has been

presented in [4]. Consequences of these attributes and metrics indicate the performance and provide base for an overall design evaluation throughout the design processes which enable to manage the software design and ultimately it helps managing the whole project efficiently and timely. Results from previous analysis of these quality attributes in earlier software designs help in managing a future projects which forms knowledge base.

To satisfy these attributes in software design for high quality software design there must be a mechanism to satisfy and manage the requirements and we will discuss a simple yet efficient framework to manage the requirements for design purpose. Fig. 4 shows the relation between the requirement management and software design components.

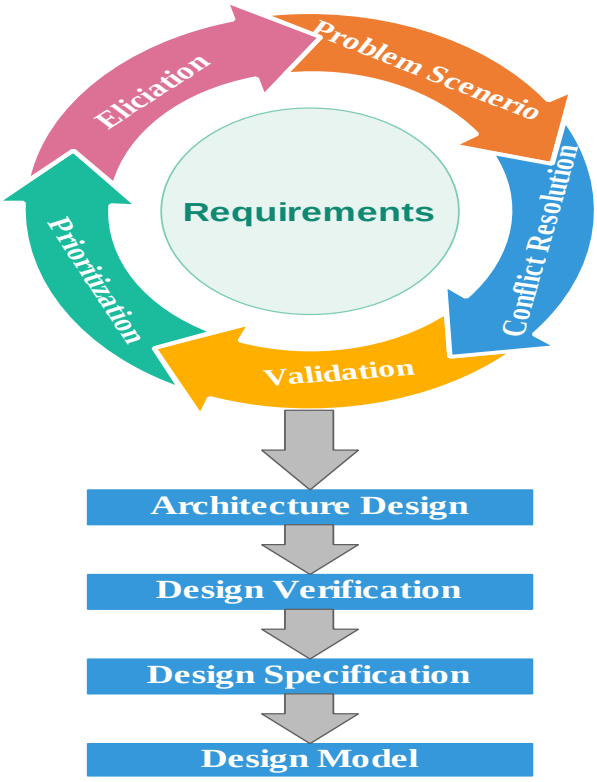

Fig. 4. Software Requirements and design relationship

According to this framework requirement management framework follows the activities in continuous fashion and validated requirements forms the basis for design process. From problem scenario requirements are gathered, filtered and any conflict in requirements are resolved by following conflict resolution phase of requirement engineering. Thus when requirements are validated upon validation a criterion, design architecture process is integrated with requirement module and software design architecture is verified against each requirement change. Whole process leads to a quality design model by following design specification which serves as a guide for designers for software design. As a whole this process is simple enough to provide validated requirements to design process continuously whenever a change is happened in requirements for the design specification and these changes are always validated in design before implementing the design yet there can be improvements in the requirement validation and prioritization phase of this framework. In our previous research on requirement prioritization we have proposed an ETVX based requirement management framework in [1]. This framework efficiently manage and prioritize requirements based on the entry, task, verification and exit criteria where at each phase requirements are validate against the entry and exit criteria and each requirement must conforms all the conditions given in exit criteria. Thus if we integrate that framework in requirement prioritization parts of this framework then it will produce better results in providing validated requirements to design phase and there will be clear understanding of design specification for the designs to follow and design the software. Main reasons to integrated our requirement prioritization model in this framework lies in the task phase of this model where HCV prioritization method is used with conflict management, requirement change management and requirement traceability which makes this model more validated to provide clear and concise requirements. Moreover it will be more efficient to use ETVX based design model for the design phase as discussed in [1].

V. TRACKING DESIGN CHANGES

As discussed earlier change is inevitable whether it is in requirement or in design. We cannot stop the changes from client but we can design our system in a way to accommodate those changes and in this scenario we need to build and follow a mechanism where every change made in design phase is documented for further reference. Frank in [7] has discussed a simple technique to track the design changes in design as shown in Fig. 5.

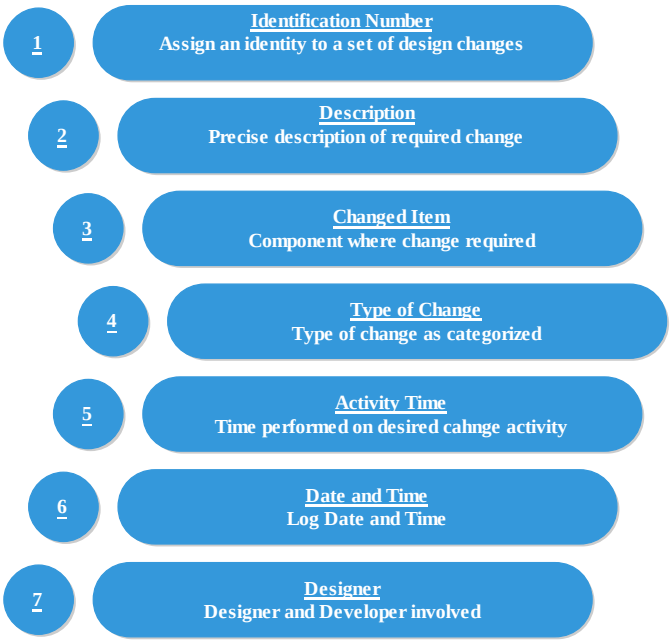

Fig. 5. Software design change tracking

Basically it's a simple and 7step model to track the design changes. It logs entries about the 7 change features as first of all any change which is recognized in design is group together with similar changes and a set is formed containing similar changes or if change is alone in its nature then single change

is considered and an identification number is allotted to that change to refer that change in change management document which is maintained throughout the design phase activity alongside the design specification. When an identification number is given to the change in design, a specific detail about the change is written which describe the change, its origin and intentions of this design change. In changed item step, relevant detail about the design component where the change is going to be made and the change itself is recorded. It's important to record the nature f change because by doing so there can be a set of changes with distinct nature. Activity time is logged to ensure the record of time it takes in implementing the change so that schedule is maintained. A detail about the time it takes, date and the particulars of developers and designers is maintained. Complete process of maintaining and tracking the design change is saved in database where all relevant personnel's can access the database and this design change tracking database serve as knowledge for future similar changes in design specifications. This way a maintained database is prepared for all the design changes and each design change is tracked based on its particulars as followed by this model.

VI. QUALITY METRIC SUIT

Quality software metrics are well known methods to assure the quality of software design and they contribute in finding defects in software design in early phases of design development. There are different quality metrics such as process metric and product metric with different goals and specific activities to assess the design, code, and test and specification quality. They help manage the project phases in a controlled and measured fashion. Today object oriented paradigm is most widely used set of models and activities lay down the basis of any software engineering project as they are most closer to reality as software engineering objects are treated as real world objects. So object oriented metrics are very helpful in measuring the quality of software design. Different techniques and methods have been proposed in literature. In this research paper we will study CK metrics in relation to assess the quality of software design and defects that are headed in design due to changes in requirement and other engineering process due to uncontrolled methods and no track of design changes.

We have already lead the foundation of requirement management and design interaction model by which prioritized and validated requirements are given to design phase whenever a change is required and all the design changes are tracked using a database comprising of essential information about design changes. Fig. 6 shows different methods included in CK metrics which work on the object oriented methodology of abstraction, inheritance, classes and cohesion. These metric methods would help us find relationship in quality metrics to find bugs and defects in design of software design by applying these techniques on object oriented interfaces revealing the quality of design and certain complexities in implemented standards ti improve the overall design.

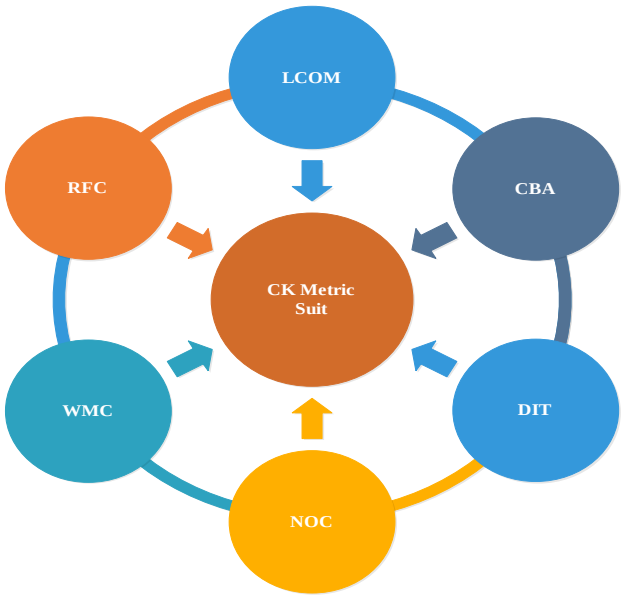

Fig. 6. CK Quality metric components

A) WMC (WEIGHTED METHODS PER CLASS)

As the name suggests, in weighted methods per class, the main idea is to count weight of methods in a class means the number of methods per class in design by using procedural methods to measure the complexity of classes under observation. Number of methods per class impacts the overall organization of class and its Childs because child of class are going to inherit its parent methods and by this way specificity is observed in class. Fig. 7 shows further sub areas of weighted methods per class.

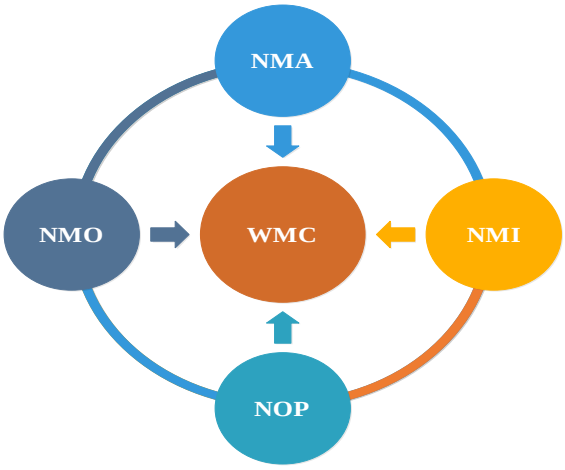

Fig. 7. Sub Categories of WMC

There are different types of weighted methods per class known as number of additional methods added, number of override methods, number of methods inherited, and number of parents of a class and this WMC method assess the overall complexity of a class because it is not wise to maintain a complex class because any addition or subtraction in later stages will cause maintainability issue.

B) DIT (DEPTH OF INHERITANCE TREE)

It is always desirable to have minimum hierarchy of inherited classes and depth of inheritance tree measure the depth of number of classes in inheritance from parent. Fig. 8 shows inheritance tree of a class.

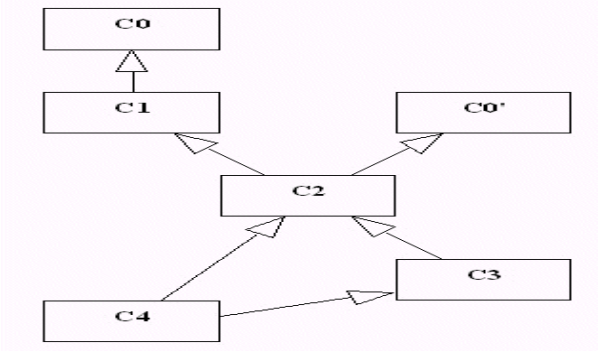

Fig. 8. An arbitrary Inheritence tree

Fig. 8 depicts a scenario where complexity is increased which increase the testing work if number of classes in inheritance tree and more and it will be difficult to maintain the class methods and inherited class relations. Maximum length path is found using DIT function to calculate the number of direct inherited classes from one parent class.

C) NOC (NUMBER OF CHILDREN)

Just like depth of inheritance tree, main idea of NOC is to calculate number of children of a specific parent class under observation because number of children effect the design and may lead to unpredictable class hierarchy where abstraction is misused. In terms of class, it calculate number of sub classes and in terms of package its number of sub packages under inheritance and its desired to have lower number of children in a class or package. There should always be a lower and upper limit on number of children in a package or a class.

D) LCOM (LACK OF COHESION METHODS)

Cohesion measures the representation of class whether multiple abstractions or single abstraction and also it define closeness of class methods and attributes. As opposed to NOC and WMC a quality program requires high cohesion in class. It counts methods in a class not sharing some instances with other methods. Increase in cohesion increases high extensibility, reusability and maintainability.

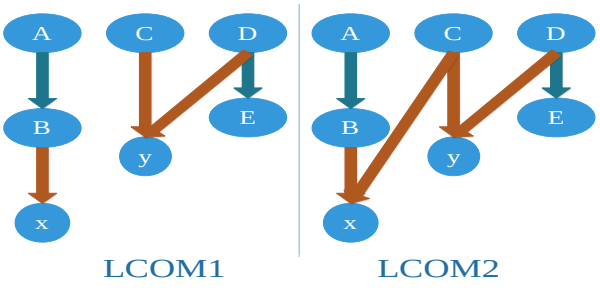

Fig. 9. cohession between modules

Fig. 9 represents basic terminology behind modules of a program depicting low cohesive and high cohesion. LCOM1 is showing less cohesion while LCOM2 class is highly cohesive because in LCOM2 method C is accessing variable x while in LCOM1 there is less interaction between methods and variables while in LCOM2 there are more interaction between methods and variables as different methods are accessed by different variables and cohesions is more than LCOM1.

E) CBO (COUPLING BETWEEN OBJECTS)

It is highly desirable that classes should be less dependent on each other because classes with high coupling are difficult to maintain and it reduces flexibility and increase complexities in class relationship which further reduce reusability because when 2 classes are tightly coupled then it means they cannot survive without each other. So CBO counts the coupling between objects.

F) RFC (RESPONSE FOR A CLASS)

It fundamentally measures the probable communication of a under observation class. By this method we get the number of methods that have been invoked in that particular class in certain condition under certain stimuli event that caused the class to invoke certain method to satisfy the request. By standard there should be sound functionality in methods of class and there should be upper limit on number of different methods invoked by one request as there should be possible functionality summed up in minimum number of methods to satisfy functionality.

VII. CONCLUSION AND DISCUSSION

After the extensive literature review by following the hybrid research model I analysed and explore various factors affecting the design of software and explored requirements as one of the factors introducing changes in software design leading to some flaws and defects in design and explored corrective actions in the form of requirement tracking and management using framework where requirements are traced and managed into design in efficient manners and we studied software quality metrics to measure the performance and correcting bad software design. If requirement gathering phase is followed efficiently then there are less chance of requirements leading to unnecessary design changes causing bad software design issues and if requirements are unavoidable then we must find a way as discussed to track the changes in design of software and take corrective actions to remedy the design by measuring and comparing the performance and quality of implemented design with the help of quality software matrices like ck metric suite.

In this research article we have concluded that changes in requirements and design can lead to poor quality design and these changes can be controlled, managed and tracked in a manner that designer can track all the changes to requirement document or in reverse from requirement to design with the help of simple tracking technique as discussed in section 5. While on the same time CK metric suite facilitate the developers and designers to build upon object oriented

methodology by continuously assessing the performance of overall design by comparing the design and code against quality attribute of software design as discussed in section in section 4 of this research article. Beside this research, many researchers has found relationship between CK metric suite design elements and Pearson coefficient to further analyse the impact of changes and to detect bad software logic and components. Numerical identification and measures are assigned to these correlations and they are plotted against the CK metric suite components and methodologies and it enable developers and designers to get more about internal structure and let them detect bad behaviour and faulty sequence early in quantitative and qualitative manners for future use. More detail about these correlations is explained in [14].

Although this research is limited to the impact of design changes due to requirement changes but yet it provides solid basis to overcome these issues by providing and introducing matrices to measure the performance of design and analysing bad software design issues. I tried to include possible research articles in my research covering the causes of design changes, impact of these changes on design leading to bad software design yet there are healthy chances that some important points might be missed. There might be some flaws in systematic procedure of selecting and interpreting the knowledge because focus was on broad topic. Also there can be biased views on this research because all the process of selecting the research articles, synthesis of data, data extraction, review and compiling has been carried out by a single researcher. In future I would like to extend this study to correlate other development activities influencing the design of software to satisfy the successful implementation and improvement with the help of a researching team with each team member with specific roles and responsibilities to be carried out to cover more knowledge.

REFERENCES

[1] Muhammad Abdullah Awais, “Requirements Prioritization: Challenges and Techniques for Quality Software Development” ACSIJ Advances in Computer Science: an International Journal, Vol. 5, Issue 2, No.20 , March 2016

[2] Nurmuliani N, Zowghi and Powell S (2004).Analysis of Requirements Volatility during Software Development Life Cycle, In: Proceedings Australian software engineering conference, pp 28–37, IEEE.

[3] Nedhal Al-Saiyd and Esraa Zriqat "Analyzing the Impact of Requirement Changing on Software Design, European Journal of Scientific Research" November 2015.

[4] Shahid Iqbal, Muhammad Khalid and M .N.A. Khan, "A Distinctive Suite of Performance Metrics for Software Design", International Journal of Software Engineering and Its Applications, Vol. 7, No. 5, pp.197- 208, 2013.

[5] B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” Vol. 2, pp. 2007–01, 2007.

[6] Elaine Barnett-Page and James Thomas, Methods for the synthesis of qualitative research: a critical review, Economic and social research council ESRC National Center for Research Methods.

[7] Frank Padberg, "Tracking the Impact of Design Changes During Software Development", University at Karlsruhe, Germany.

[8] Maddeh, M,Ghedira and Romdhani, "Classification of Model Refactoring Approaches”, Journal of Object Technology" 8(6):143-158

[9] Briand, L., Melo, W. and Basil," A Validation of Object Oriented Design Metrics as Quality Indicators", IEEE Transactions on Software Engineering, Vol. 22, pp. 751-761, 1996.

[10] Chidamber, S.,Darcy, D., and Kemerer, C.F., Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis. IEEE Transactions on Software Engineering, Vol. 24, No. 8.

[11] Hidamber, S., Kemerer, C, “A Metrics Suite for Object Oriented Design”, IEEE Transactions on Software Engineering, Vol. 20, No. 6, 1994.

[12] Bansiya J. and Davis. C.G., “A Hierarchical Model for Object Oriented Design Quality Assessment”, IEEE Transactions on Software Engineering, 2002.

[13] Vipin, S. And Kumar, S. "Impact of Coupling and Cohesion in Object Oriented Technology, Journal of Software Engineering and Applications, Vol. 5, No. 9, pp. 671, 2012.

[14] Kuljit Kaur Chahal, Hardeep Singh, “Metrics to Study Symptoms of Bad Software Designs”, SIGSOFT Software Engineering Notes, Vol. 34, No. 1, January 2009.