ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 7 N. 3 (2018), 31-45
eISSN: 2255-2863
DOI: http://dx.doi.org/10.14201/ADCAIJ2018733145

# An empirical approach for software reengineering process with relation to quality assurance mechanism

Muhammad Muzammul[a] and Dr. M. Awais[b]

[a]Department of Software Engineering, GCUF
[b]Department of Software Engineering,GCUF
m.muzammul275@gcuf.edu.pk, muhammadawais@gcuf.edu.pk

KEYWORD

*Software; reengineering; refactoring; restructuring; forward engineering; reverse engineering; quality assurance; internal quality; external quality; flexibility; reusability; reliability; robustness*

ABSTRACT

*Software development advances focus on productivity of existing software systems and quality is basic demand of every engineering product. In this paper we will discuss complete reengineering process with aspects of forward, reverse and quality assurance mechanism. As we know the software development life cycle (SDLC) follows a complete mechanism of engineering process. In forward engineering we tried to follow selective main phases of software engineering(data,requirements,design,development,implementation). Inreverse engineering we move backward from the last phase of developing product as it gather requirements from implemented product(implementation, coding, design,requirements,data).During reengineering we add up more quality features on customer demands, but the actual demand is to fulfill quality needs that can be assured by external as well as internal quality attributes such as reliability, efficiency, flexibility, reusability and robustness in any software system. We discussed a methodological approach to move from reengineering to the journey of quality assurance. More than 50 studies come into discussion and throughput results proposed by graph and tabular form. We can say if the reengineering process produce quality attributes, then it can be said by old software system refactoring as code refactoring, data refactoring and architectural refactoring we obtained a quality products at lower cost instead of new software system development, which causes decrease in quality attributes as cost, time etc. In future work testing methodology can be proposed for quality assurance.*

## 1. Introduction

The re-engineering [1] [13] of used things in already developed software lead to avoid from wastage of material, time and maintainace cost bring most of effects on economic values [4]. For any software development organizations [2] increase in software development cost is much important, so the development companies which are at small, medium or large scale try to follow up reengineering process.

Basically, the re-engineering process is to get existing software and add more features according to customer requirements [3]. The factors effects in these fields are software maintenance cost, repairing cost or the

performance of system architecture failed to work out. We try to move with technology advancements and available hardware [5]. However, the main problem comes into existence when we try to understand current system. Usually, the documentation with architectural design and source code is not present or expired due to large time passage so we cannot get its view clearly [12]. What we have to do here? Usually present features not needed, we exclude old features with many more advance features according to customer demand [6].

Re-engineering also involved sub process as forward engineering, reverse engineering, refactoring or re-structuring [7]. Reverse engineering process is the major workout of software reengineering. We check implemented software system and analyze its activity by moving the implemented organization. We study the system and extract its coding by its working or its documentation [8].. We conclude the architecture from designing phase analysis, after this phase we get extract the modules and get requirements as the system was developed.

Before Further explanation of intro

## 1.1. Methodology

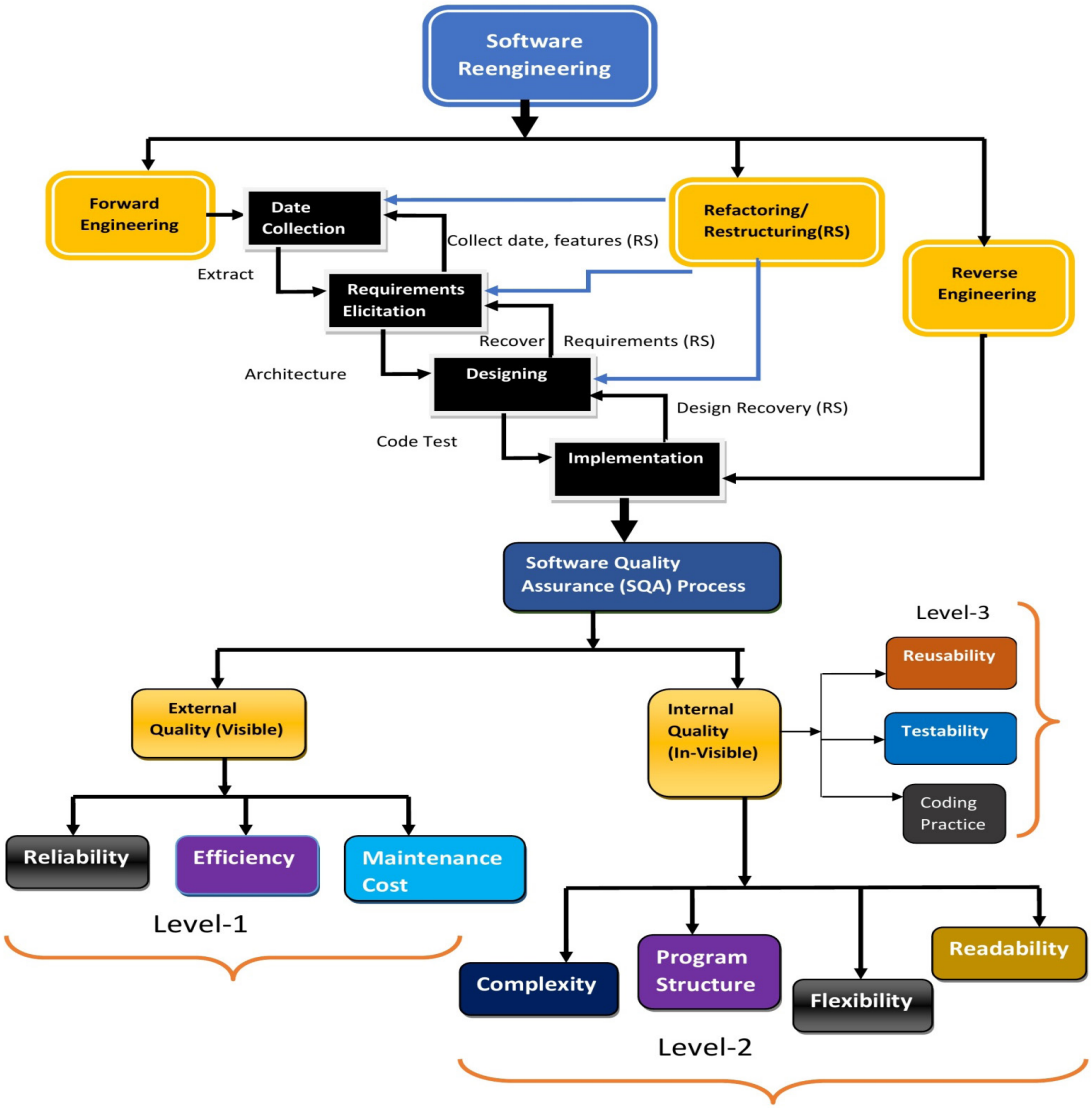
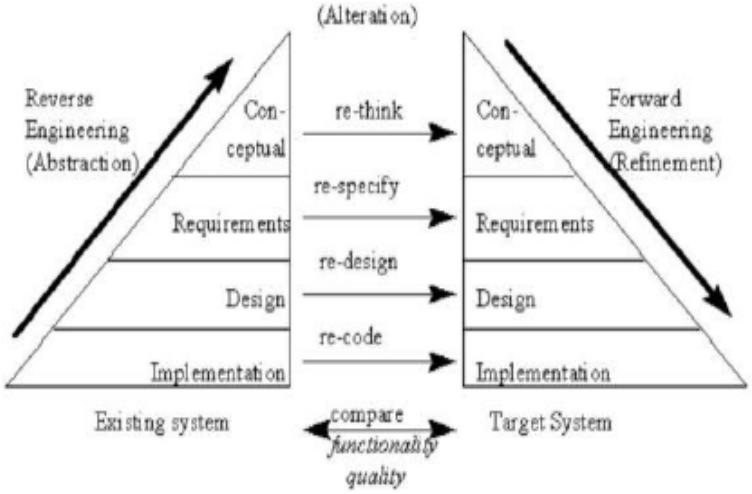



*Figure 1: Model for software reengineering to quality assurance [46][48][49][22]*

In gathered data we add more features according to customer demand. We structure a document as collected form of data [16]. After this we start to develop new system or we can say restructured system.

In forward engineering the reengineering process phase refactoring [14] involved equally. We get elicitated requirements from data or we can say requirements are refactored [19]. After requirements elicitation we add up more designing architectural features in existing architecture that is called design restructuring [9]. From newly developed design we develop and refectories the code [17] [18]. We check coding logics, errors, coding smells, and all drawbacks that were lagging the quality [10].




Software system refactoring complete here and we can say reengineering [11] is a method to enhance the quality attributes in system with code refactoring, design refactoring and software refactoring and prepare new product with extra features and more reliability. Forward engineering last phase is testing to implement the software system [15]. We test and evaluate system according to customer demands and here refectory product ready to implement [20] [21].

In this paper I proposed next procedure to assure the quality of newly reengineered software.3-level architecture is proposed for the testing quality of developed software [22]. Level-1 which tells us about the external quality which is visible to customer .In external quality there are three attributes reliability, efficiency and maintenance cost [23]. These attributes are visible part to our customer these should be present at least 90% for the customer satisfaction. In level-2, 3 I formulated the invisible part of reengineered system as internal quality. Internal quality at level-2 tells that there should be improvements in code quality [25], decrease in complexity and increase in user friendly as readability and programming structure should be well organized and in level-3 here in our reengineered software there should be presence of reusability [24]. Test ability process should be easy to evaluate the quality. In any future dis-orderness if we have to study the code it should be structured in well professional practices [26] [27] [28]. At last, I evaluate that if we will follow my proposed methodology we can consider it easy mechanism from the journey of software reengineering to quality assurance.

## 2. Related Work

This section provides a discussion on reengineering process to quality assurance characteristics. It also illustrates the forward engineering, reverse engineering and quality assurance process with the help of refactoring process applied to deal with code bad smells and design patterns refactoring. A brief discussion on existing reengineering approaches is also covered in this section.

## 2.1. Software re-engineering

The process of extracting any artifact with added features, for enhanced performance and reliability high degree of consistency and betterment in maintainability is called re-engineering [29].

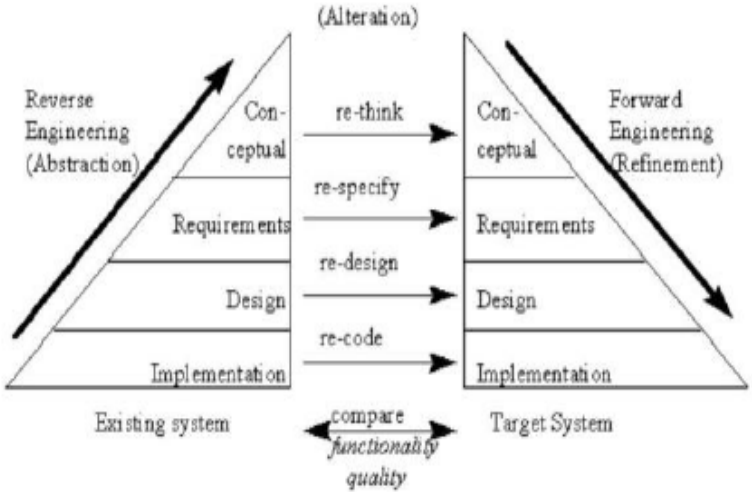



*Figure 2: the general pattern of re-engineering software [45]*

Now a days software products advancements increasing rapidly. Mostly software developed with new architectures attributes and technologies did not work well as the old legacy systems provide user attachments capabillities.Re-engineering provide facility of user boost advance software capabilities with the reuse of existing resources [11]. Reengineering process in large-scale legacy software with the changing in interfaced can be risky. When we increase requirements integrity, it leaves effects on security. By software elicitation tried to overcome these risks of software integrity [12].

## 2.2. Restructuring (code, data, design, document)

Re-structuring process can be said refactoring in real meaning, it can be defined as" Refactoring is a well-organized method for restructuring an existing body of code, make changing in its internal structure- without changing its external behavior [30].

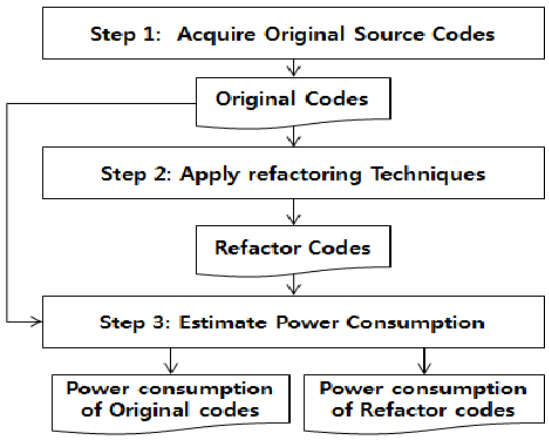



*Figure 4: Code refactoring [47]*

On-chip BRAM becomes highly important for high-bandwidth data communication. Automated chip restructuring is best practice to elaborate the buffering and bandwidth control. It checks impact on the performance and the consumption of resources [17].

Restructuring in IT areas as customization, internal process, deployment of software, also organization changing is discussed. IT served quality, Customer satisfaction and user impact on IT projects. [18].

To improve quality there is need to detect and remove the errors in working system environment as in form of code or architecture. Several tools and techniques can be used for the betterment of code quality, design quality and overall system quality. As the quality of code/design will be good then the quality of software product will automatically be good [9-10].

Refactoring and restructuring improve the reliability and maintainability of code. The main purpose is the identification of potential refactoring opportunities. Terms used here 1) refactoring 2) replace type code within subclasses 3) Replace code type with state. Mostly focused on Java and also on automatically refactoring methods [31].

## 2.3. Forward and reverse engineering with aspects of refactoring

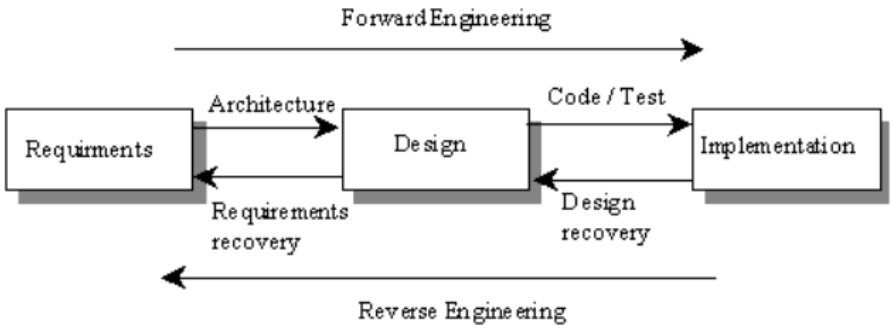



*Figure 3: Reverse and forward engineering [46]*

### 2.3.1. *Reverse-Engineering*

(Implementation, design, requirements, data)

It is the process of analyzing software system to extract the design, requirements, and data from the implementation of system with high level of abstraction is called reverse engineering [32].

Reverse Engineering (RE) in the semiconductor business. It has connected different systems, for example, the electronic administration, enormous picture speeding up calculation, and programmed age for circuit extraction. Because of leading the RE with the self-made ROIC chip, it was affirmed that the simple circuit was precisely extricated as the entryway level [33].

### 2.3.2. *Forward Engineering:*

(Data, requirements, design, implementation)

It is the process of engineering software by following steps from data to extract design for architecture with implementation of coding and system [32].

An appropriated programming item building group needs to manage the extra issue of dissemination separated from the typical desires around cost, quality, and time to market and advancement. Disregarding exclusively following the recommended programming building forms, regularly the circulated groups neglect to go about as a solitary item group. The key speculation in this approach is the suspicion that most dispersed programming item designing groups in a similar association requires arrangement as opposed to base up retooling as a detailed programming building activity and this arrangement can be accomplished in a quick and successful way by adjusting the key interface pioneers [34].

## 2.4. Reengineering connection with Quality assurance

Our research question that can be extracted to fulfill the conclusion of all the research is based on the quality assurance. Here we will try to explain the concepts of

1) Quality assurance
2) Software quality dependences
3) Software Reengineering and quality assurance
4) Software quality attributes w.r.t software forward and reverse engineering

### *What is Software Quality assurance?*

Software quality assurance (SQA) is a method of testing software that our developed product fulfilling the quality specification standards and compile and developed according to rules.SQA is a running process of (SDLC) that checks developed software system working according to desired quality measures [35].

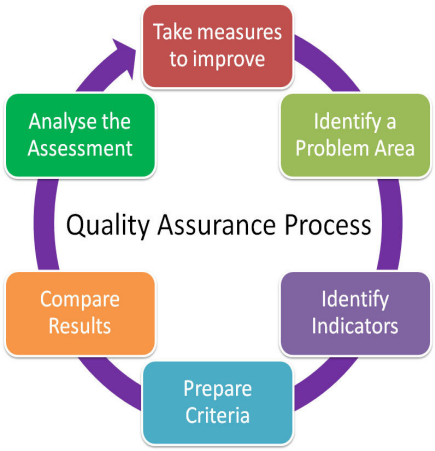



*Figure 4: Process of quality assurance [47]*

### *How Software Reengineering leads to quality assurance?*

As in Figure1.We proposed a model which is giving complete idea of re-engineering toward quality assurance. It can be said as old software systems re-engineered we get new product with advance features. If we follow development standards, we obtained a new product with more and advance features.

Programming configuration designs were elevated to make the plan of projects more "adaptable, measured, reusable, and reasonable". We at that point set out to examine the effect of configuration designs on various quality properties and distributed a paper entitled "Do Design Patterns Impact Software Quality Positively?" In this review paper for the honor, we report and consider our and others' investigations on the effect of configuration designs, talking about some key discoveries detailed about plan designs [36].

I actualized and ran my first clone recognition on modern programming approximately 10 years prior. From that point forward, our examination models have developed into a business apparatus utilized by proficient programming designers around the globe consistently. Every one of us only work on, or utilize as a major aspect of our review administrations, programming quality examinations based upon this present group's exploration [37].

### 2.4.1. *Methodology for SQ improvement*

The software procedure display is utilized to guarantee software quality, speak to an assortment of assignment settings, oversee venture length, enhance the procedure and range to execute the procedure understanding, and to suitable verifiable guess for all undertaking settings. Given this perspective, this paper shows another software improvement life cycle display, "AZ-Model", for software advancement by presenting new exercises amid software advancement life cycle [38].

### 2.4.2. *Design patterns and Quality Assurances*

The nature of software frameworks relies upon a few elements and one of them is the means by which the software planners utilize the outline designs in the outline of software [39].

Code quality issues can cause serious problem. Before going to in depth programming there is need to get perfect skills for code quality. Students should follow the techniques, there should be a flow in code, and issues can be accruing for code quality. Modularization and decomposition can be caused. If students investigate these faults, timely then can use tools to solve the problem [40].

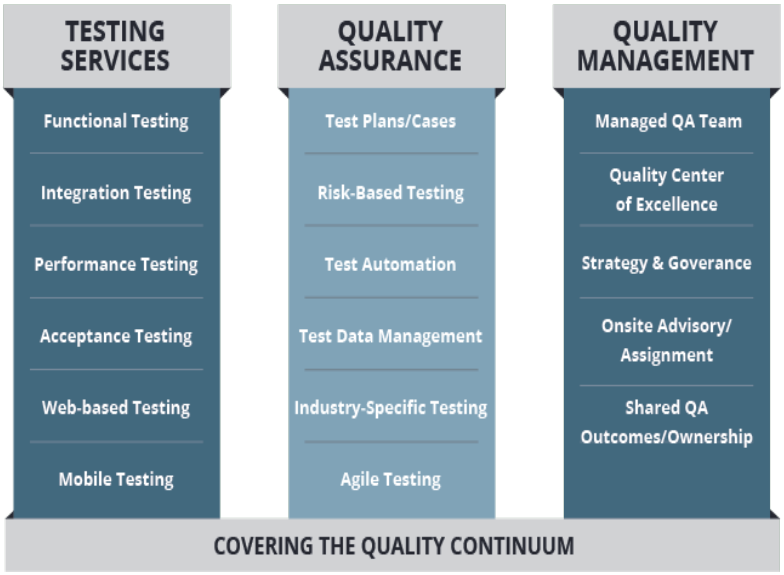



*Figure 5: Process of quality assurance [50]*

*2.5 Tabular representation of Old related word with field of reengineering to quality assurance*

| Authors | Issues Found | Methodology Applied | Major Output/Finding | Ref# |
|---|---|---|---|---|
| CharilaosPetrou (2016) | Signal processing with big data | principal component analysis (PCA) | Big data can be processes by different signal processing techniques. | [41] |
| Jason Cong, PengWei, CodyHao Yu (2107) | High level synthesis fail to put arrays design on-chip | Automated on-chip buffer restructuring can resolve the issue | Our automated source-to-source code transformation tool improves the performance of a broad class of high level synthesis | [17] |
| Fernando Szimanski, Anivaldo S. Vale (2018) | IT have to fulfill customer requirements | Business models and analyzing tools | IT provides services quality, customer satisfaction and engagement, as well as transparency on IT projects. | [18] |
| KashyapTodi, Jussi Jokinen (2017) | Computational problems exists in software due to layout and designs | Human System Visualization | Well organized software systems with user friendly interfaces are developed | [19] |
| Amit Rathee. (2017) | Due to increase in features and advance capabilities , we restructure software | Cohesion techniques are used to rebuilt the classes of OOP based system | Well designed, organized, restructured, new systems are obtained after restructuring | [20] |
| Nathan ManeraMagalhães. (2017) | Unnecessary structural complexity may occur, in which a program has a cycloramic complexity | Identify complexity, restructuring code and finding flow graphs | The approach is able to support unnecessary cycloramic complexity remove | [21] |
| JyothiVedurada. (2017) | Large volume of code causes complexities | Replace code with sub classes | Restructured well designed and user friendly system is developed | [42] |
| HiekeKeuning (2017) | Code quality is a big issue in these days | Functions clearity, expressions finding demoularization | Clear and refactored with clear quality programs are obtained | [40] |
| Ana Rodriguez (2017) | Mobile rely on batteries and APPS use much battery | Code refactoring reduce power consumption | Via code restructuring new coded apps reduced the power usage | [43] |
| Didier Rémy (2017) | Inductive data types and parametric polymorphism are two common problems | Inductive data-types and parametric polymorphism | By adding or dropping some parts of codes then it is possible to make automated pointing system that can be helpful in | [44] |

# 3. Background

This section provides a discussion on software re-engineering process with quality characteristics.

## 3.1. Software re-engineering

A variety of software reengineering process to quality characteristics has been reported in literature such as: Forward engineering, reverse engineering, refactoring (code, data, and design) according to Bhatnagar [48] Re-engineering is the only way to utilize the software fully and solve the problem of software backlogs . Software re-engineering may be the only viable way to ensure that legacy systems can continue in service [36] by Foutse Khomh. It may be too expensive and too risky to adopt any other approach to system evolution.

To understand the reasons for this, we must make a rough assessment of the legacy system problem [35]. R. Dewar in 1999 tells [51] reengineering of legacy systems is a method that has great importance and still significantly resisting the process of modification and evaluation for the purpose of business goals which are constantly changing. J.Clarke in 2003 [52] Metaheuristic techniques such as genetic algorithms simulated annealing and tabu search have found wide application in most areas of engineering. H. Jaakkola 2010 [53] software design and development coexist and co-evolve with quality provision, assessment and enforcement. However, most and also modern research provides only bread-and-butter lists of useful properties without giving a systematic structure for evaluating them. Rajesh H. Kulkarni [54] recently, the modeling of whole process of software (SW) development is performed using extended waterfall and agile models. For the development of any software we mostly use software development life cycle, literature guides us lot for this process: Rosa E. Quelal [55].Agile methodologies have been increasingly used in software development projects worldwide. However, there is little information about the adoption of these methodologies in Latin America. Michael Kirchhoff 2018 [56] Faster development of new algorithms is crucial in modern projects. Highly abstracted, data flow and modular oriented model-driven development methods and tools are used for this purpose. Asim Iftikhar 2018 [57] Global software development is an example of the modern age.

Team members can split work in different modules, can communicate with each other due to boundaries in physical appearance and time availability factor effects. Different software development companies are scattering their work at national as well as international level. In Forward engineering we start working from getting data and from passing processes requirements elicitation to designing and till the end implementation we obtain a quality product.

### 3.1.1. *Code refactoring*

Software refactoring meaning that we transform software code or design in such a way that it improves the working quality of software while behavior remain preserved [58]. Opdyke proposed several techniques of refactoring ath design and implementation stages of software development [59].We can elaborate refactoring process with several steps related with source code and a model developed. These process steps was proposed by Wake [60] at starting history, the advancement was done byMens and Tourwe [61].In general point of view we follow these steps for code refactoring

1- Find software refactoring parts
2- Slect appropriate approach for refactoring
3- Check preservation of behavior
4- Apply the approach selected for refactoring
5- Analyzing the refactoring impact on quality of software
6- Ensure the presence of consistency in code and UML Models

### 3.1.2. *New Code refactoring Methodology*

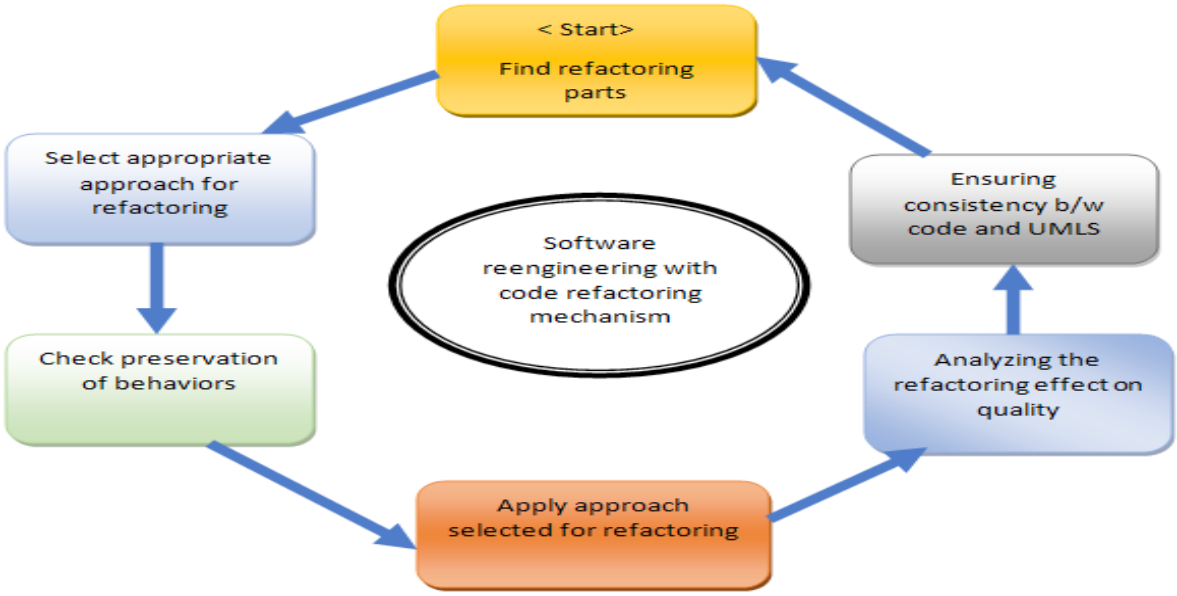



*Table 2: surveyed detection techniques for code refactoring*

| Author(s) | Software metrics | Bad smells | Tool |
|---|---|---|---|
| Arendt and Taentzer [64] | Coupling, Cohesion, Inheritance | Concrete super class, long parameter list, equal attributes in sibling classes, unused use case, create associated classes, pull up attribute, introduce parameter Object. | EMF Smell and EMF Refactor |
| Fourati et al. [65] | Coupling, Cohesion, Complexity, Inheritance | Blob, Lava Flow, Functional Decomposition, Poltergeists, and Swiss Army Knife | No |
| Moha et al. [66] | Coupling, Cohesion, Complexity, Inheritance | Blob, Functional Decomposition, Spaghetti Code, Swiss Army Knife | DETEX |
| Ghannem et al. [67] | Coupling, Cohesion, Complexity, Inheritance | Blob, Functional decomposition, Poor use of abstraction | Rule generation |
| Van Gorp et al. [68] | Coupling, Cohesion, Inheritance | Pull up method, extract method | MDA CASE |
| Ruhroth et al. [69] | Cohesion, Inheritance | Hidden Concurrency, Unnecessary Behavioral Complexity, Too Low Cohesion, Too strong coupling, refused bequest | RMC |
| Saeki and Kaiya [70] | Complexity | Not specified | No |
| Mohamed et al. [71] | Complexity | Blob | MREFACTOR |
| Jensen and Cheng [33] | Coupling, Complexity, Inheritance | Abstract access, delegation, encapsulate construction, partial abstraction. | Metrics detection tools |
| Enckevort [31] | Coupling, Cohesion, Complexity, Inheritance | God class, cyclic dependency, abstract class, poor use of abstraction, encapsulate field, long parameter list, data class. | AndroMDA |
| Kempen et al. [72] | Coupling | God class | SAAT |

*Table 3: Code refactoring tools report*

| Tool | Smell detection | Type | Code linkage | Language support |
|---|---|---|---|---|
| Checkstyle [74] | Duplicated code, large class, long method, long parameter list | Eclipse, standalone | Yes | Java |
| Décor [75] | Data class, god/large class, long method, long parameter list, message chain, refused bequest, speculative generality, tradition breaker. | Standalone | No | Java |
| iPlasma [76] | Brain class, brain method, data class, duplicated code, extensive coupling, feature envy, intensive coupling, refused bequest, shotgun surgery, tradition breaker | Standalone | No | C++, Java |
| inFusion [7] | Brain class, brain method, data class, data clumps, duplicated code, extensive coupling, feature envy, intensive coupling, refused bequest, shotgun surgery, tradition breaker | Standalone | No | C, C++, Java |
| JDeodorant [77] | Feature envy, god/large class, long method, switch statements | Eclipse | Yes | Java |
| PMD [78] | Dead code, duplicated code, large class, long method, long parameter list | Eclipse, standalone | Yes | Java |
| Stench blossom [79] | Data clumps, feature envy, large class, long method, message chains, switch statement, typecast | Eclipse | Yes | Java |

*Table 4: Quality assurance and quality control [81]*

| | Quality Assurance | Quality Control |
|---|---|---|
| Monitors, Improves and / or Audits: | • Document Control<br>• Document Change Control<br>• Calibration<br>• Gage R&R<br>• Corrective Action<br>• Auditing<br>• Systems Interaction Map<br>• Quality Objectives<br>• Training<br>• Preventive Maintenance<br>• Job Descriptions<br>• Purchase Order Process<br>• Preventive Action<br>• Quality Plans<br>• New product introduction<br>• Quality Management Review<br>• Failure Mode Effect and Analysis<br>• Contract Review<br>• QA Org Chart<br>• Risk Management | • Identification and traceability<br>• Non Conforming Material Control<br>• Final Inspection<br>• Receiving Inspection<br>• Process Inspection<br>• Shipping Inspection<br>• Statistical Process Control.<br>• Quality Records<br>• Raw Material Control<br>• Finish Goods Control<br>• Product Reliability<br>• Material Review Board<br>• Control Plans |
| Examples: | • Walkthrough<br>• Testing<br>• Inspection<br>• Checkpoint review | • Quality Audit<br>• Defining Process<br>• Selection of tools<br>• Training |
| Used for: | • Product<br>• Reactive<br>• Line function<br>• Find defects | • Process<br>• Proactive<br>• Staff function<br>• Prevent defects |

# 4. Conclusion

Software re-engineering purpose is to add more features in existing software system and increase its quality parameters. In our existing software environment as day by day technology going to improve then it become difficult to develop software again and again. So, re-engineering provide a mechanism by which we can improve

existing systems with disturbing the behaviors of system quality can be improved. Software re-engineering follow some steps with the help of reverse and forward engineering and according to users requirements new features are added to existing product. The parameters that can improve the quality of software can be coding improvements designing improvements and all of these requirements can be done by restructuring of data. Software re-engineering process undergo with some steps with relation to forward engineering and reverse engineering. We refractor the parameters during reverse engineering process. New product obtained with advance quality parameters. There are different levels of quality assurance which are need of system. After the implementation of re-engineering process we assure these parameters which included as two different terms quality assurance (QA) and quality control (QC). The advancements can be tested if these parameters assuredly present in our developed product. It may include reliability, efficiency, consistency, integrity, robustness, maintenance cost, complexity, programming structure, reuseability and testability.

## 5. Acknowledgment

This research paper was prepared with the support of Govt.College University Faisalabad, Pakistan in department of software engineering with the support of Dr.Awais and Mr.Yahya Saeed. And **Principal author**:Muhammad Muzammul(33103-3379246-3)

## 6. References

[1] Olexandr Kharchenko; "*Optimization of software architecture selection for the system under design and reengineering*"; 14th International Conference on Advanced Trends in Radio electronics, Telecommunications and Computer Engineering (TCSET); IEEE; 2018; Pages: 1245-1248

[2] Bernhard Dorninger; "*Reengineering an industrial HMI: Approach, objectives, and challenges*"; IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER); 2018; Pages 547-551

[3] Jaswinder Singh; "*Identification of requirements of software reengineering for JAVA projects*"; International Conference on Computing, Communication and Automation (ICCCA); 2017; page 931-934; IEEE Conferences

[4] Ana Maria da Mota Moura; "*Awareness Driven Software Reengineering*"; IEEE 25th International Requirements Engineering Conference (RE); 2017; Pages 550-555; IEEE Conferences

[5] Grace Park; "*A modeling framework for business process reengineering using big data analytics and a goal-orientation*" 2017 11th International Conference on Research Challenges in Information Science (RCIS); Pages: 21-32 ; IEEE Conferences

[6] James. J. Mulcahy; "*Reengineering autonomic components in legacy software systems: A case study*"; 2017 Annual IEEE International Systems Conference (SysCon); Pages: 1-7; IEEE Conferences

[7] VivekBhatnag," *Study of Software Development Using Software Re-Engineering*", International Journ al of Engineering Trends and Applica t ions (I J ETA)-Volume 3 Issue 2, Mar-Apr 2016

[8] Nilesh Jadav; "*How To Reverse Engineer Using Advanced Apk Tool*"; March08 2017

[9] Arcelli, F.; Tosi, C.; Zanoni, M.; Maggioni, S.: "*The MARPLE project: A tool for design pattern detection and software architecture reconstruction*." In: 1st International Workshop on Academic Software Development Tools and Techniques (WASDeTT-1) (2008) Google Scholar

[10] GhulamRasool, Zeeshan Arshad," *A Lightweight Approach for Detection of Code Smells*" Arabian Journal for Science and Engineering, February 2017, Volume 42, Issue 2, pp. 483-506|

[11] A. CathreenGraciamary , Dr. M.Chidambaram , "*EESRM: An Effective Approach to Improve the Performance of Software Re-Engineering"* ISSN 0973-4562 Volume 13, Number 6 (2018) pp. 3648-3654

[12] K.R. Wallace," *Safe and secure: re-engineering a software process set for the challenges of the 21st century*" 9th IET International Conference on System Safety and Cyber Security (2014), 2014 page 5.2.2

[13] P. Hunter." *Re-engineering data storage*" Volume 8, Issue 12, December 2013, p. 58-62

[14] J. Clarke , J.J. Dolado , M. Harman , R. Hierons "*Reformulating software engineering as a search problem*", Online ISSN 1463-9831, Volume 150, Issue 3, June 2003, p. 161-175
[15] A. Egyed, N. Medvidovic." Component-based perspective on software mismatches detection and resolution" Volume 147, Issue 6, December 2000, p. 225-236
[16] K. Rafique, Chunhui Yuan," *Re-engineering spectrum management policy framework: Meeting challenges of future",*4th IET International Conference on Wireless, Mobile & Multimedia Networks (ICWMMN 2011), 2011 p.35-39
[17] Jason Cong, Peng Wei, CodyHao Yu" *Bandwidth Optimization through On-Chip Memory Restructuring for HLS*" Austin, TX, USA – June 18-22, 2017
[18] Fernando Szimanski,Anivaldo S. Vale," *Restructuring Information Technology Area: an experience report in the public service*", A Computer Socio-Technical Perspective-Volume 1, Pages 63, Goiania, Goias, Brazil – May 26-29, 2015, Tokyo, Japan – March 07-11, 2018
[19] KashyapTodi, Jussi Jokinen;"*Familiarization: Restructuring Layouts with Visual Learning Models*", 23rd International Conference on Intelligent User Interfaces, Pages 547-558,
[20] Amit Rathee," *Restructuring of Object-Oriented Software Through Cohesion Improvement Using Frequent Usage Patterns*" Software Engineering Notes, Volume 42 Issue 3, July 2017 Pages 1-8,
[21] Nathan ManeraMagalhães," *An Automated Refactoring Approach to Remove Unnecessary Complexity in Source Code*", Systematic and Automated Software Testing, Article No. 3,Fortaleza, Brazil – September 18-19, 2017
[22] Software quality assurance; Visited website 25 June, 2018; https://www.testinstitute.org/What_is_Software_Quality_Assurance.php; International software testing institute;
[23] External vs internal quality; Visited website 25 June, 2018; https://meekrosoft.wordpress.com/2010/10/31/internal-and-external-software-quality/
[24] Emile Swarts; "*Internal vs External Quality of Software*"; written on 29th September, 2015 tagged in Ruby on Rails, Software Architecture
[25] Jehad Al Dallal; Anas Abdin. "*Empirical Evaluation of the Impact of Object-Oriented Code Refactoring on Quality Attributes: A Systematic Literature Review*".IEEE Transactions on Software Engineering (2018), Volume: 44, Issue: 1pages: 44-69
[26] John Jagtiani; Christian Bach; Chris Huntley. "*Leveraging Big Data From Open Source to Improve Software Project Management*". IEEE Engineering Management Review (2018), Volume: 46, Issue: 1Pages: 65-79
[27] Robert Chatley; Lawrence Jones. "*Diggit: Automated code review via software repository mining*".IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (2018): pp. 567-571
[28] Luigi Franzio; Bin Lin; Michele Lanza; Gabriele Bavota; "*RETICULA: Real-time code quality assessment*". IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (2018): 542-546
[29] VivekBhatnag," Study of Software Development Using Software Re-Engineering", International Journ al of Engineering Trends and Applica t ions (I J ETA)-Volume 3 Issue 2, Mar-Apr 2016
[30] Martin Kropp, University of Applied Sciences Northwestern Switzerland," MSE-SEA-Restructuring.pptx". Available at: https://wiki.hsr.ch/MasterModulSEA/files/MSE-SEA-Restructuring.pdf (Accessed: 10 April 2018).
[31] JyothiVedurada," Refactoring opportunities for replacing type code with state and subclass", ICSE-C '17, Pages 305-307, Buenos Aires, Argentina – May 20-28, 2017
[32] Parminder Singh, "Software reverse engineering",Student at Punjabi University, https://www.slideshare.net/parrychahal50/software-reverse-engineering-69635162, Published on Nov 29, 2016,
[33] Gyungtae Kim; Ming Ma; Inhag Park, "A fast and flexible software for IC reverse engineering", International Conference on Electronics, Information, and Communication (ICEIC), 2018, Pages: 1-4,IEEE Conferences
[34] Bhaskarjyoti Das; "An empirical approach for optimizing globally distributed software product engineering";2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI);Pages: 1340-1348

[35] Software Quality assurance visited 15 April 2018, https://www.techopedia.com/definition/4363/software-quality-assurance-sqa
[36] FoutseKhomh; Yann-GaëlGuéhéneuc. "Design patterns impact on software quality: Where are the theories?". IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (2018): 15-25
[37] ElmarJuergens. "A decade of software quality analysis in practice: Surprises, anecdotes, and lessons learned (keynote)".IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (2018): 1-1
[38] Muhammad Azeem Akbar; Jun Sang. "Improving the Quality of Software Development Process by Introducing a New Methodology–AZ-Model". IEEE Journals & Magazines (2018): 6, 4811-4823
[39] Muhammad Noman Riaz. "Impact of software design patterns on the quality of software: A comparative study". International Conference on Computing, Mathematics and Engineering Technologies (2018):1-6
[40] HiekeKeuning," Code Quality Issues in Student Programs", ITiCSE '17, Pages 110-115, Bologna, Italy – July 03-05, 2017
[41] Charilaos Petrou," Signal Processing Techniques Restructure the Big Data Era" Article No. 52, Patras, Greece – November 10-12, 2016
[42] JyothiVedurada," Refactoring opportunities for replacing type code with state and subclass", ICSE-C '17, Pages 305-307, Buenos Aires, Argentina – May 20-28, 2017
[43] Ana Rodriguez," Reducing energy consumption of resource-intensive scientific mobile applications via code refactoring", ICSE-C '17, Pages 475-476, Buenos Aires, Argentina – May 20-28, 2017
[44] Didier Remy,Inria, France," More automated code refactorization and code reuse (invited talk)", Haskell 2017, Pages 1-1, Oxford, UK – September 07-08, 2017
[45] Byrne, E.J., A Conceptual Foundation for Software Re-engineering, in Conference on Software Maintenance 1992
[46] Sivaram; "*The Myth of Software Reengineering*"; Posted On December 24, 2013 by GB Shah filed under Programming
[47] Jae Jin Park; "*Investigation for Software Power Consumption of Code Refactoring Techniques*"; Published 2014 in SEKE; Page 717-718
[48] Manzoor Ahmad Rather; "Study of Software Development Using Software Re-Engineering"; March 2016; page 53-54
[49] http://www.bmtechsolutions.com/software-quality-assurance/; visited 3 july 2018
[50] Serena Josh; "Top 10 Challenges as a QA Analyst Tester in Software or Web Development"; http://www.zarantech.com/blog/top-10-challenges-qa-analyst-tester-software-web-development/
[51] R. Dewar ; A. D. Lloyd ; "Identifying and communicating expertise in systems reengineering: a patterns approach"; June 1999, p. 145-152
[52] J.Clarke; "Reformulating software engineering as a search problem"; June 2003, p. 161-175
[53] H. Jaakkola; "Framework for high-quality software design and development: a systematic approach" ; April 2010, p. 105-118
[54] Rajesh H. Kulkarni; "Integration of artificial intelligence activities in software development processes and measuring effectiveness of integration"; February 2017, p. 18-26
[55] Rosa E. Quelal; "A survey of agile software development methodologies in Ecuador"; 13-16 June 2018; IEEE
[56] Michael Kirch
hoff; "Increasing Efficiency in Data Flow Oriented Model Driven Software Development for Softcore Processors"; p. 23-27 July 2018; IEEE
[57] Asim Iftikhar; "A survey of soft computing applications in global software development"; 11-12 May 2018; IEEE
[58] M. Fowler, K. Beck, J. Brant, W. Opdyke, Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999.
[59] W.F. Opdyke, Refactoring Object-Oriented Frameworks, University of Illinois at Urbana-Champaign, 1992.

[60] W.C. Wake, Refactoring Workbook, Addison-Wesley Professional, 2004.
[61] T. Mens, T. Tourwé, A survey of software refactoring, IEEE Trans. Softw. Eng. 30 (2004) 126-139.*
[62] P.P. Stepan Cais, Identifying software metrics thresholds for safety critical system, The Third International Conference on Informatics Engineering and Information Science (ICIEIS2014), The Society of Digital Information and Wireless Communications, 2014, pp. 67-78.
[63] D. Bhalla, Automatic Detection of Bad Smells in Java Code, California State University, Long Beach, 2009.
[64] T. Arendt, F. Mantz, G. Taentzer, EMF refactor: specification and application of model refactorings within the Eclipse Modeling Framework, Proceedings of the BENEVOL Workshop, 2010.
[65] R. Fourati, N. Bouassida, H.B. Abdallah, A metric-based approach for anti-pattern detection in uml designs, Computer and Information Science 2011, Springer, 2011, pp. 17-33.
[66] N. Moha, Y.-G. Gueheneuc, L. Duchien, A.-F. Le Meur, DECOR: a method for the specification and detection of code and design smells, IEEE Trans. Softw. Eng. 36 (2010) 20-36.
[67] A. Ghannem, M. Kessentini, G. El Boussaidi, Detecting model refactoring opportunities using heuristic search, Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research, IBM Corp., 2011, pp. 175-187.
[68] P. Van Gorp, H. Stenten, T. Mens, S. Demeyer, Formal UML Support for the SemiAutomatic Application of Object-Oriented Refactorings, University of Antwerp, Citeseer, 2003.
[69] T. Ruhroth, H. Voigt, H. Wehrheim, Measure, diagnose, refactor: a formal quality cycle forsoftware models, 35th Euromicro Conference on SoftwareEngineering and Advanced Applications, 2009. SEAA'09, IEEE, 2009, pp. 360-367.
[70] M. Saeki, H. Kaiya, Model metrics and metrics of model transformations, The First Workshop on Quality in Modeling, 2006, pp. 31-45.
[71] M. Mohamed, M. Romdhani, K. Ghedira, M-REFACTOR: a new approach and tool for model refactoring, ARPN J. Syst. Softw. 1 (4) (2011) 117-122.
[72] M. Van Kempen, M. Chaudron, D. Kourie, A. Boake, Towards proving preservation
[73] Eclipse, EclipseHomepage. Available: https://www.eclipse.org/, 2016 (accessed 23 February 2016).
[74] Checkstyle.https://checkstyle.sourceforge.net/, 2016(accessed 23February2016).
[75] Décor. https://www.ptidej.net/download, 2016 (accessed 23 February 2016).
[76] iPlasma. https://loose.upt.ro/reengineering/research/iplasma, 2016 (accessed 23 February 2016).
[77] JDeodorant. https://github.com/tsantalis/JDeodorant, 2016 (accessed 23 February 2016).
[78] PMD. https://pmd.sourceforge.net/, 2016 (accessed 23 February 2016).
[79] Stench Blossom. https://github.com/DeveloperLiberationFront/refactoring-tools/ wiki/Stench-Blossom, 2016 (accessed 23 February 2016).
[80] G. Soares, R. Gheyi, T. Massoni, Automated behavioral testing of refactoring engines, IEEE Trans. Softw. Eng. 39 (2013) 147-162.
[81] Visited on 3 july2018 http://jobsandnewstoday.blogspot.com/2013/04/what-is-quality-assurance-quality-control-testing.html.