Model Driven Re-engineering with the Fields of Re-structuring: Software Quality Assurance Theory

Muhammad Muzammul

Department of Software Engineering, GCUF, Pakistan

DOI: 10.29322/IJSRP.8.6.2018.p7861
http://dx.doi.org/10.29322/IJSRP.8.6.2018.p7861

Abstract- Due to developing trend for the reliability of software with new and advanced features, reengineering is core demand of this age. For the purpose of assurance of proposal validity, a huge analysis of existing approaches with proper referencing performed and proposed a new generalized model based theory. In this paper, phases of reengineering as core of restructuring and refactoring, reverse and forward engineering is going to be discussed with the objective of quality and user requirements. Restructuring covers the areas as code patterns, object-oriented patterns, an architecture for the extraction of design and restructured documents generated. By the restructure of data, an advanced product with more features can be obtained by passing all steps of forwarding engineering. Code smells removing with the improvement in reliability and user trust refactoring process provided great help. If user required quality features needed to add in existing software we perform reverse engineering. This theory suggested a generalized approach (fig2) with all types of interconnections among the areas of re-engineering and provided a solution by testing user needs according to proposed requirements that are quality definition. Future work leads to comparison among all techniques of code restructuring with requirements elicitation terms, and a general proposed approach needed.

Index Terms- Reengineering, Restructuring, Reverse Engineering, forward Engineering, Code Restructuring, Data Restructuring, Quality Assurance, Rearchitecturing, Testing, User Requirements, Requirements elicitation, Designing Architecture, Code Restructuring, Design restructuring

I. Introduction

Advancement in software engineering fields and daily developments of software project in this area, a lot of techniques and processes introducing knowledge in this field. Old software performed a Lot of contribution for the user's needs, but advancement in technologies and with the increase in user quality requirements, a generalized approach is need of hour. Due to increasing demand in this area, new products by adding some features in old products there is need to introduce. Reengineering provide an instructional behaviours, trends and analysis to fulfil users needs. Sometime we restructure code, design, data and documents, the need of all these activities to give new product with more features. Several phase of reengineering were already proposed (fig1).A new a organized form of model we have tried to furnished. The overall process covering several areas of software reengineering [1].All phases are interlinked. The proposed model is an hierarchal approach with all phases of forward engineering as data restructuring extract user requirements in elicitated way. Then architectural techniques applied to extract required design and then implemented with the testing methodologies according to user needs. This process remain in working with forward as well as reverse engineering phases and user demanded product with quality assurance obtained. In fig2. Flow of model represent the aspects Reengineering(RE),Restructuring(RS),Forward engineering(FE),Forward Engineering(FE) with phases, Data restructuring(D-R),requirements elicitation(REl),designing(RS),implementation as (DR,REL,RS,Impl) and Reverse engineering(RvEng) with phases(Impl,RD,REL,D-R) leads to software testing(ST).The study is based on more than 100 similar proposed papers. Each aspect is elaborated with the help of proper referenced and author's theory. We used 10-15 papers for the explanation of each point and area of reengineering. For each area and process of reengineering a table is proposed which explains the issues, methodologies, inputs and outputs. A hierarchical Model is represented in (fig2) which leads to step by step process of Software reengineering toward quality assurance .High level-1 is Reengineering and level-2 is forward engineering, reverse engineering and restructuring.level-3 is legacy system/pattern restructuring and customer demand testing, here it is tested that after reengineering process is the system is full of customer demand if satisfies then after testing we say it is assuring the quality. If testing not fulfilling customer demands it is moved back word to level-2 for again process of forward engineering. Process remains in flow till the required software system is proposed.

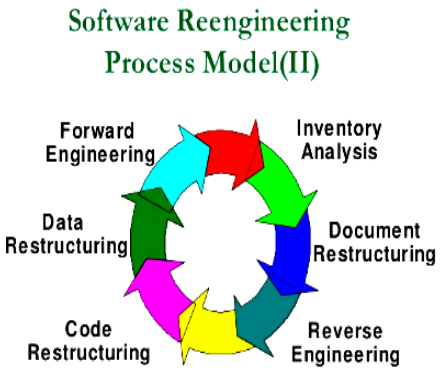


Fig1.Phases of Reengineering Process [1]

1) Reengineering process and analyzing reengineering technique from different systems:The process of extracting any artefact with added features, for enhanced performance and reliability high degree of consistency and betterment in maintainability is called re-engineering". [1]

1.1) Human being and Machine Interfacing:

Human beings and machine interfacing (MI) plays a central role for the working of several machines. Various factors like cost pressure, change in subsequent requirements, validity of requirements leads toward reengineering the interfaces for valid change in designs patterns, change in generation human interaction. It also depends upon the and advancements in technology. Several automated techniques for each module are used for the adoption of advances. Here author tried to evaluate reengineering MI framework on the report of statistical analysis of existing source code, objectives, challenges, transferring from source code to required components, existing product and configuring with full-fledged maintain and configuration of reengineered product.[2]

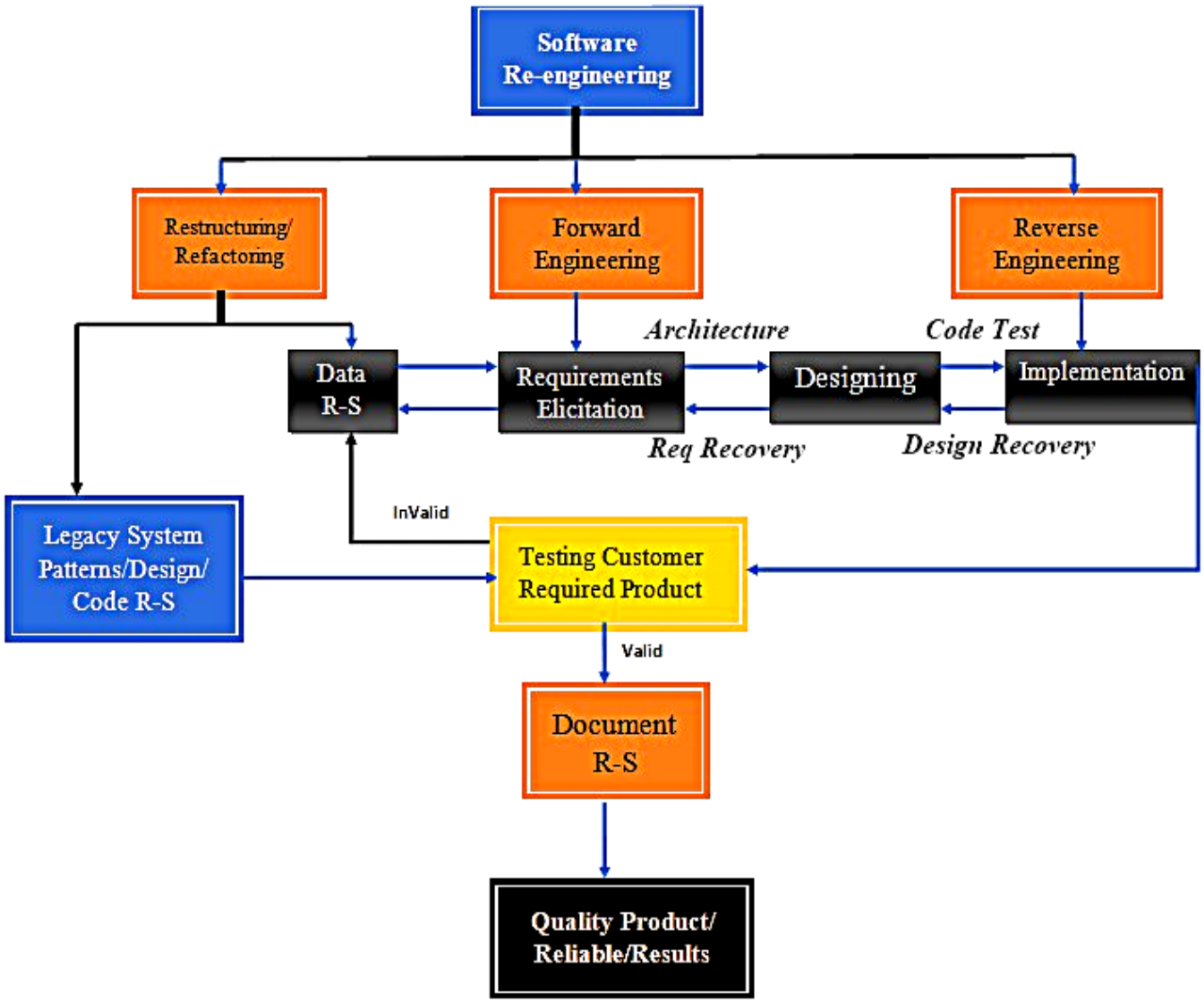


Fig2.Model Based Reengineering Process w.r.t QA. [1]

1.2) Code smells findings and refactoring: By the detection and removal of errors in code, it is easy to improve the quality of software. Several techniques can be used for the betterment of code quality. As the quality of code will be good then the quality of product will automatically good. Author proposed several comparisons and combined 22 different methods. Flexibility of the proposed solution can be reasonable because it is implemented on different eight open source software. [3] [4]

1.3)Business Process Reengineering (BPRE): Organizations needed to add up best process in their running environment for the achievement of their goals. Many methods can be helpful for BPRE process patterns can be helpful for situational methods reengineering (SMRE).The proposed

solution provides the inter connection between BPRE and SPRE.A framework added to existing BP can be helpful to increase software business throughput. [5]

1.4) Improving re-engineering performance

Now a days software products advancements increasing rapidly. Mostly software developed with new architectures attributes and technologies.Re engineering provide facility of user boost advance softwarecapabilities. Here new method of reengineering is proposed to resolve previous issues. Datais converted here in bytes and are compared with data sets. As the result, the access time of our system is very small as compared to any other data types because these are converted into bytes and easy for computing. [6]

1.5) Safe and secure process in 21th century: 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. Analysis applied on systems for the maintenance of integrity .A template is proposed for risk elicitation and other type of risks like acquisition/supply chain, legal s well as human effects. [7]

1.6) Re-engineering of data storage: Storage technology relating advancements becoming popular day by day. It is almost done by pros and cons with the comparison of solid-state drive and hard disk. It is becoming a big challenge for the applications, which are using intensive amount of data, and it is need to analyze massive data. [8]

1.7) Raising software engineering as a search problem

Met heuristic techniques (as genetic algorithms, simulated annealing and tabu search) have wide range of applications in business, financial as well as engineering fileds.In SE it is used for test data generation, clustering of software modules and for the prediction of costs. Several problems in SE yet not solved. Met heuristic search applicable in most of SE problems. Features that enable applicable also discussed in paper. Areas like maintenance/evolution system integration and requirements scheduling of our SE discusses using met heuristic search. This search also provides solution of many software engineering or re-engineering problems that are constraints based. It rises SE as a search problem and try to overcome these problems. [9]

1.8) Software components mismatch detection and resolution: It is big problem for development and implementation of software modules, many of components mismatches during project. This search provide analysis of resolving such raised problem to get better quality product.[10]

1.9)Managing wireless networks spectrum and reengineering

ICT becoming a word level communication perspective, wireless technology leave an impact on economic rate. Spectrum management framework and technology emerging needs has been discussed. Stakeholders and emerging technologies also affect the spectrum. [11]

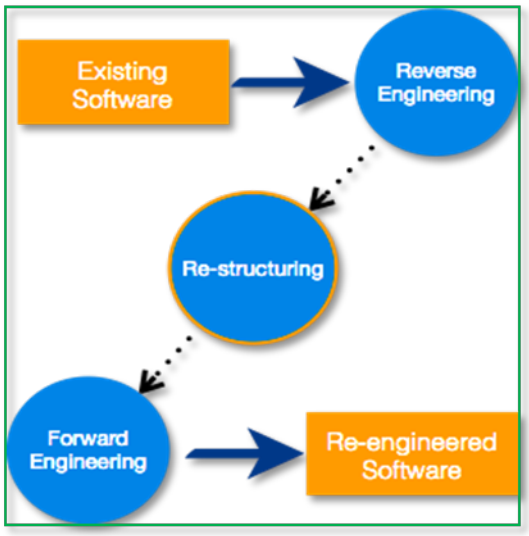
1.10) Object oriented Based legacy systems

(Fig3.Explaining the concept of re-engineering to restructuring)[28]

Due to new technologies, software products changing day-by-day architectures and attributes.OO (object oriented) was used mostly as base of legacy systems. These systems needs to improve object baseness due to poor methods like inheritance, so reengineering demand increases, many techniques are used for this purpose.

II. RE-STRUCTURING LEAD TO QUALITY FEATURES FROM PREVIOUS TO ADVANCE PRODUCT

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".[13].It exist in following areas



Following papers will elaborate the concept of above-mentioned diagram.

2.1) Signal processing Method for restructure

Big data mainly concentrate on organizations, academia and government. Main purpose is to get information from big amount of data with is heterogeneous. Signal processing technique is related to many statically analysis and principal component analysis (PCA).Many concepts comprehensive sampling (CS), Convexoptimization (CO).In some cases (KBT) are used for robustness, comprehension and reduce problems that arises in big data.[14]

2.2) on chip memory restructure for bandwidth

High-level synthesis (HLS) leads to high quality and high-productivity designs. 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. We find out problem non-linear programming integer (INLP) and can be solved by hardware description language (HDL).The conclusion we obtained, automatedsource-to-source code transformation improves the performance. [15]

2.3) Effect on Information technology field

IT area is to supply hold up so effective and efficient that the finalistic areas can achieve their missions, goals, indicator and observance with permissible requirements. Information systems based should be analyzed for originating the product for business goals and objectives. In this discussion mainly focused on 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. [16]

2.4) Layouts and visual learning Models

There is large variation in features due to differences in designs. We try to find out common features at same location. Computational approach describes the restructuring layouts. New product with advance features can be find out quickly.Familiarisation inspired with human visual system(HVS) that can be helpful for automate the design for users.[17]

2.5) Re-structuring object-oriented software

Herepattern techniques applied on cohesion improvement. Toproduce well-designed software according to customer requirements object oriented programming adopted. Cost reduced and reusabilityincreased. Without disturbing external functionality internal code of system refectories. Refactoring also refer to remove code smells and improve cohesion and reduce coupling. Frequent used patterns work for this term (FUP).Software refactored on the base of algorithm, selectedcluster, source code refactoring depends upon selected algorithm. [18]

2.6)Automated refactoring approach to remove code complexities

Incorrect code is difficult to test and maintain. Unnecessary complexity may accrue due to code smell. Thisapproaches suggest to refractor code to identify complexities and suggest to programmer to refractor the code through controlSource code automated refactoring is need of the research and able to solve the problems and also suggesting the concept of unit testing.[19]

2.7) Replacing code type with state and subclasses

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.[20]

2.8) Issues in quality of code

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. Modularization effects cannot be solved completely by tooling methods.[21]

2.9) Scientific apps via code refactoring

Mobile Grids and Mobile-edge clouds lead toward new computing area. With growing capability and scientific methods make mobile attractive. Mobiledepends upon batteries for power consumption. Code refactoring can reduce the consumption of energy. Herepurposed several code-refactoring opportunities. [22]

2.10) automated code refactoring and reuse

Parametric polymorphism and inductive data types widely related to structure programming. Haskell and ML are more correct methods by code construction. Coderefectory and code reusable are two different concepts. These can be helpful by automatically type in-consistent program leads to incomplete pattern machings.Posteriori program abstraction rule help for code transformation. On the ornamented structure some adding, dropping parts of code can be helpful for refactoring. Barcode is similar to genetic lifting and can be worth full for adoption. [23]

2.11) Relationship between refactoring and change

In refactoring of any system, we perform internalchanging without changing external behavior. This approach provides us benefits in form of code quality andproductivity. Number of changing performed in requirements rather than software quality. Study based on surveys different developers. This technique focus on different types of techniques as Fault Repairing Modification, Feature Introduction Modification, and General Maintenance Modification. Developers also work on refactoring in improving maintainability and comprehensibility of the source code when fixing bugs, refactoring operations included cohesion. New features are introduced in product. [24]

2.12) Impact of refactoring on code smell

Software refactoring can remove structural problem. Smelly structure because low quality.13 types of code smells are addressed for different 23 projects. Some time refactoring tends to produce more code smells. A percentage based result proposed. [25]

2.13) Software restructuring Model-based

Due to increase in volume of software industry, thereis needed to restructure the code to control cost. Correct methodology needed due to inflexibility of restructuring tools. To make tools accurate we discussed with software developers. Amodel driven interface developed. Compilation COM interfaces used. High-level model driven code proposed to transform low-level code. Generic-integrated technology used. Automatic restructuring patterns also discussed. This approach given a contribute in software development. [26]

2.14) SRT-restructuring tool for java software

Software move in several maintenance steps during its working. Best development practices and design patterns can support. This result move toward low rate of cohesion and high coupling with low maintainability.STR leads to packages instead of classes, by using combinational heuristics. [27]

III. Reverse Engineering with aspects of data restructuring and forward engineering as in fig1.3

Basic areas of work

1) LegacySoftware systems
2) Reverse engineering
(Implementation, design, requirements elicitation)
2) Restructuring (Code, design, data)
3) Forward engineering
(With requirements, newdesign, and implementation)
4) Testing with quality assurance techniques (customer demand)

5) Validity/Invalidity leads to product or Forward engineering
6) If valid leads to quality assurance otherwise back to the process again
7) After testing quality assured product obtained

Explanation with the help of already works.

Legacy software system reengineering leads to quality software with previous and advance capabilities

3.1) Legacy software systems reengineering?

Such type of software that was working since long time and still used for business needs are known as legacy software. When we use the work legacy, software systems then we called both hardware/software. With the advancement in technology and software engineering field the life of old systems decreases. To full fill advance business needs these systems need to reengineer. Basic purpose is to improve quality and add more advance features. It can be involved new hardware as well as softwares.Legacy software systems are everywhere in Banks, nuclear stations, TV stations, manufacturer industries, Companies of energy, powergeneration. When we will reengineer these systems, we can move forward with this advance technology age in every field of life. [29]. Concept explanation with other's theories.

3.2) Automatic components reengineering in legacy systems

Modern large-scale organizational software developed with one or many legacy systems that fulfill demands of offline and online clients. Due to customer needs, the systems often implemented with complexity. For the purpose of order processing, security screening, shipping related tasks these systems communicate with internal or external legacy components. Within passage of time, stakeholders demand to connect with other legacy systems using different data communication techniques. Systems should be designed in such a way that can be easily modified without adoption of complexity. It is therefore useful for solutions for the individual components to be designed with an architecture that lend themselves to be easily reengineered or replaced without disturbing the other system components, and without adding significant complexity to the overall system of systems. In this case, study the need of operators, troubleshooters tried to eliminate, in future reengineering becomes easy. The system is tested using different data communication protocols with coded programming language than original implementation. The study is demonstrated with original automatic design patterns. [30]

3.3) Managerial dimensions on reengineering process

Instead of functionality here managerial issues as market competition, operational, organizational and financial constraints addressed. Economic strategy of existing legacy systems, effectiveness of cutthroat product analysis plan, financial support for the progress task, disturbing market factors, development process and effect of cutthroat product on cost & quality of target system. [31]

3.4) Software reengineering supported by database reverse engineering w.r.t legacy systems

Legacy systems with old databases front end COBOL we based resulting complexities in performances. We reengineer the systems with advance features of database management systems using reverse engineering process. We develop high-level categorized model driven approach. [32]

3.5) Legacy systems reengineering to metaprogramming

Model-driven development (MDD) is subjected area in software development. Its basic purpose is to transform the designs, implementations, architectures of legacy softwares.This approach leads reengineering of legacy systems with the help of MDD and metaprogramming. In Reverse Engineering leads to object-oriented model based on legacy application code and database and, in Forward Engineering, the object-oriented model is developed and used as basis to meta-programs perform code generation. [33]

3.6) Refactoring Cost Estimation (RCE) Model

Successful software systems are need of hour. Object-oriented systems built as any legacy system. We identifying many reengineering patterns, which capture best practices in reverse and reengineering object-oriented systems. To, make old systems more maintainable reengineering applied. Refactoring is one type of re-engineering. Cost refactoring is RCE.These caused by class misuse, violation of the principle of encapsulation, lack of use of inheritance concept, misuse of inheritance, misplaced polymorphism. [34]

3.7) Software architecture for power market

Large-scale software architecture is proposed for web based software systems that provide support for economic solution in situation of irregular power market. That is useful for various power market structures and can be extended for fast changing in industry interfaces. Wrapping technique work in this situation for the power market supporting software systems. [35]

3.8) Reengineering risk quantify for legacy system

Our software is asset of any organization due to large number of risks. The success depends upon the working of legacy systems and solution of these systems problems. Effective reengineering is necessary to evaluate these risks. Here risk measurement model is presented for components materialize to check comprehensive impact on legacy systems. We can decide about evaluation of legacy systems. [36]

3.9) Ways of evaluation in legacy systems

Old systems with previous techniques were called legacy systems. These systems have greater business values for several organizations. In performance, these systems considered week due to weak requirements engineering process. Due to non-flexible structures, these systems are considered difficult to maintain with changing requirements. Hangings in environment made these systems complicated. [37]

3.10) Software reengineering for DSP-based systems on a chip

An expanded multifaceted nature of present day PC based frameworks is joined by cutting edge programming designs inserted into a framework. Regular approach for outlining customer electronic items incorporates use of inheritance code with a few adjustments keeping in mind the end goal to run it on the focused on stage. Shorter improvement time and spending imperatives are additionally affecting the procedure of framework outline. Since each implanted framework is having its particular impediments, reuse of installed programming without adjustment to the new stage is not generally conceivable. This paper depicts a contextual analysis of programming reengineering of Dolby Virtual Speaker from the reference stage to the required DSP construct framework with respect to a chip,

including framework check and framework approval continuously conditions.. [38]

3.11) Requirements reconciling via reengineering

In present day programming improvement, programming prerequisites and usage are not generally accommodated. In particular, for present day setting mindful frameworks, changes in partners' prerequisites reveal that current execution is lacking to meet the new necessities because of the confinements of the customary strategies. In this paper, we propose a setting mindful necessities elicitation way to deal with accommodate the hole between programming prerequisites and execution for setting mindful administration development in light of a novel reengineering system approach. [39]

3.12) Legacy escaping safety in critical embedded system

Sort safe abnormal state dialects, for example, Java have not yet discovered their way into the area of profoundly installed frameworks, despite the fact that various endeavors have been made to influence these dialects to cost alluring. We display an approach that permits to isolated consolidate inheritance programming segments and safe programming segments in an installed framework utilizing the two most basic correspondence figures of speech found in this area. We exhibit the achievability of our approach by porting a non-paltry piece of a genuine, hard constant inserted aeronautics application. Our outcomes demonstrate that the cost of this blended mode activity is on an indistinguishable scale from the unadulterated task.[40]

3.13) Reengineering of enterprise software for cloud computing

Distributed computing is the future pattern for big business programming arrangements, which implies many inheritance frameworks should be either adjusted to fit the prerequisite of distributed computing or to be cleansed and upgraded without any preparation. Once the cosmology is worked, there will be a connection amongst metaphysics and endeavor programming. Thirdly, the organization of big business programming metaphysics is done through the product re-engineering exercises. Once the metaphysics is worked, there will be a connection amongst cosmology and endeavor programming. By examining the ideas and relations in metaphysics, the undertaking programming will be comprehended and deteriorated as various administration competitors.[41]

3.14) IT Infrastructure Reengineering

ICT frameworks are made of programming, middleware and equipment segments and are normally appropriated over a system. We propose a reengineering strategy to find the topology of a conveyed IT framework, in view of a multinomial calculated relapse demonstrate and an arrangement of topology generalizations. To exhibit the practicality of the approach we connected the model to a few associations with conveyed ITIs and, among different viewpoints, we found that the most intermittent generalizations are the brought together and spine ones. The topology of that ITI postures imperatives on programming calculations, information structures and programming setup, because of concerns, for example, adaptation to non-critical failure, idleness or synchronization.[42]

IV. Reverse Engineering and Forward Engineering

Overview: These are phases of reengineering where we obtain product with advance features. The process of reverse engineering move in phases as (implementation, design, requirements, data restructuring) and in forward engineering we move from the phases as (data, requirements, design, implementation)

Why we need?

We need to get data from implemented system as customer is addicted old system interfaces but

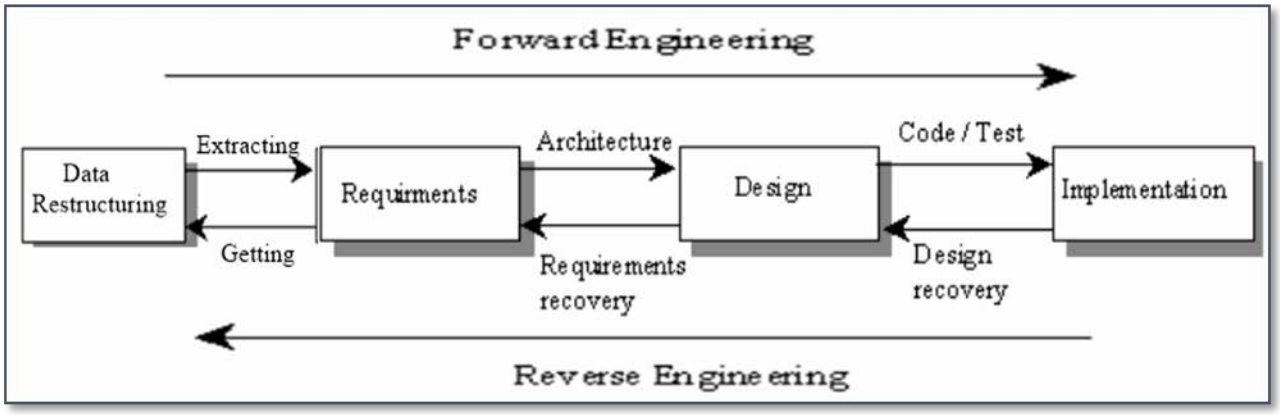


Fig.5.Forward and reverse engineering.[43]

Want latest engineering technologies implementation, so we need to design new system with advance and more features that were already in old system. We move reverse and then by adding more features for new product we move forward and design advance and latest software system, whichare configured according to latest technologies, interfaces, and computing hardware as well as software.

Reverse-Engineering (Implementation, design, requirements, data)

It is the process of analyzing software system to extract the design, requirements, data from the implementation of system with high level of abstraction is called reverse engineering.[44]

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.[44]

Explanation of Concept with already done work by different author's reference

4.1) Reverse engineering and open source software

Program investigation is the procedure of statically or powerfully recovering the structure and conduct of programming frameworks. Static investigation exclusively depends on the accessibility of source code of PC programs, while dynamic examination catches program data-utilizing execution follows amid program runtime. In this paper, we appear and examine how open source programming ventures altogether add to the advancement, development, and approval of program examination systems and improving figured out graphs with valuable and important data, notwithstanding for those methods that depend on powerful investigation.[45]

4.2) Software for IC Reverse Engineering

This paper introduces a quick and adaptable CAD apparatus for 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.[46]

4.3) Conventional Software Reverse Engineering

In contemporary years, joining among investigate territories of semantic web innovations and programming building occurred because of the reason of designers being available at various virtual, social, and land areas. Because of this amalgamation, another worked together field has developed known as Semantic Web Enabled Software Engineering. This field presents analysts abundant chances to test issues and difficulties, which are started because of their amalgamation. This exploration paper shows a system and examines the usage way to deal with take steps to the above issue.[47]

4.4) Trace analyzer for real time software

With the accessibility of the AUTOSAR standard, show driven systems are getting to be built up in the car area. In any case, the way toward making models of existing framework parts is regularly troublesome and tedious, particularly when the inheritance code must be re-utilized or data about the correct planning conduct is required. Keeping in mind the end goal to handle this figuring out issue, we display Cortina, a novel apparatus that infers an AUTOSAR agreeable model of a constant framework from a dynamic investigation of its follow accounts.[48]

4.5) Process mining event log analysis

To comprehend and keep up the conduct of a (heritage) programming framework, one can watch and concentrate the framework's conduct by dissecting occasion information. For show driven figuring out and investigation of framework conduct, activity and utilization in view of programming occasion information, we require a mix of innovative calculations and methods. In this paper, we introduce the State chart Workbench: a novel programming conduct investigation apparatus. Our apparatus gives a rich and develop combination of innovative (scholastic) strategies for the examination of conduct, execution (timings), recurrence (utilization), conformance and dependability about different formal models. They went with Eclipse module enables the client to intelligently connect every one of the outcomes from the State chart Workbench back to the source code of the framework and empowers clients to escape with their own particular programming.[49]

4.6)Hierarchical analysis recursion modeling

This paper presents 1) a novel order and recursion expansion to the procedure tree model; and 2) the principal, recursion mindful process display disclosure strategy that use various leveled data in occasion logs, regularly accessible for programming frameworks. This strategy enables us to break down the operational procedures of programming frameworks under genuine conditions at various levels of granularity. Trial comes about in view of genuine living (programming) occasion logs exhibit the plausibility and value of the approach and demonstrate the tremendous potential to accelerate revelation by abusing the accessible chain of importance.[50]

4.7) Hierarchical analysis recursion modeling

This paper presents 1) a novel chain of command and recursion expansion to the procedure tree model; and 2) the primary, recursion mindful process demonstrate disclosure strategy that use progressive data in occasion logs, commonly accessible for programming frameworks. This strategy enables us to break down the operational procedures of programming frameworks under genuine conditions at different levels of granularity. Test comes about in light of genuine living (programming) occasion logs exhibit the practicality and handiness of the approach and demonstrate the tremendous potential to accelerate disclosure by abusing the accessible progressive system.[51]

4.8)Sentimantics programming languages and software **support**

These days, software engineering progressively utilizes formal strategies to improve comprehension of complex programming frameworks and to reason about their conduct regarding a formal particular. Since the semantics is a fundamental piece of a formal meaning of a programming dialect, we have arranged a bundle of modules, that assistance us and to understudies to comprehend the most prevalent semantic strategy - auxiliary operational semantics. The main module deciphers a program written in a programming dialect to digest machine code, the second module influences turn around interpretation from code to program to source content and the third one copies stepwise execution of unique machine code.[52]

4.9) Software clustering hill climbing

Bunching methods are utilized for separating programming design in figuring out process. Extricating the Call Dependency Graph (CDG) from the source code is the initial phase during the time spent programming bunching. This chart is firmly coupled to the utilized programming dialect with the goal that the current toolsets for developing a CDG takes a shot at the specific programming dialect. Along these lines, utilizing existing CDG extraction toolsets for expansive scale programming frameworks, e.g., Mozilla Firefox, which composed by various programming dialects, is incomprehensible. The SDG can be free of programming dialects, consequently, serves to the product specialist to grouping the expansive scale programming frameworks to remove programming design, meaning to comprehend and keep up the current programming frameworks.[53]

4.10)Code obfuscation and Reverse Engineering

Figuring out is the procedure of decompiling and dismantling the executables to recoup the source code/get together code installed inside it. While figuring out is the way toward inspecting the code, in hostile setting the aggressors can re-design the code, which prompts programming robbery? The fundamental thought is to disguise the exclusive code area by utilizing preventive plan muddling and addition of self-adjusting code at the parallel level. The mix of configuration level confusion and the addition of self-changing code changes over the code into a semantically identical one that makes it hard to figure out. The test comes about evaluate the level of muddling, stealth of the code, and impacts on execution time and code estimate.[54]

4.11) Software code smell and multiple versions

The side effects, which mirror the poor outline nature of code, are known as code smells. Refactoring is one of the conceivable approaches to evacuate code smells, yet refactoring does not want allowed to engineer. For this reason an observational examination on dissemination of various code smells over various forms of ventures is given in this paper, with the goal that refactoring systems can be constructed keeping in a view what smell is more viable and at which time amid development of programming. b) God smell and earnest infringement have more commitment than different scents though examples having Type Checking smell are less in every one of the adaptations.[55]

4.12)Fuzzy framework for network reverse engineering

Fluffing is a basic piece of secure programming advancement life cycle, for discovering vulnerabilities, creating adventures, and figuring out. In this paper, an efficient approach is proposed and apparatus model created for the digital red joining purposes. For a situation examine, the created Buzz apparatus is utilized to figure out a restrictive NATO Link-1 arrange convention permitting to infuse rebel plane tracks into air tasks charge and control framework.[56]

4.13) Reverse engineering awareness

While reengineering a framework for redesign and recovery, an association genre-partner reassesses how the framework actualizes abnormal state business prerequisites and makes alterations to fit in with wanted changes. The greater part of the exertion on self-versatile framework reengineering is centered on division of concerns (i.e.: adjustment rationale and application rationale) or develop a non-versatile to a self-versatile framework. We recognized that the outline recuperation approaches being utilized, restrain themselves to recoup the application rationale, consequently leaving the adjustment rationale in low-level models. We comprehend that recouping the adjustment operationalization and mapping them onto based objective model will give an abnormal state portrayal of both the adjustment rationale and the application rationale. [57]

4.14)Using abstract syntax tree Source code to sequence diagram

It is important for software maintenance. Extract-abstract-present model can also be used in engineering process. In this situation source code is converted into specific structure.AST (abstract syntax tree) can also be used to extract structure. Here the process of reverse engineering from source code to sequence diagram is discussed. It is concluded that AST assist in reverse engineering process.[58]

4.15) Using reverse engineering and rapid prototyping for patient

Orthotic gadgets are as of now intended to fit a scope of patients and subsequently they don't give individualized solace and capacity. A mechanized method for creating tolerant particular orthotic gadgets can possibly give incredible solace and permit to changes in the standard outline to meet the particular needs of every patient. A novel procedure was designed to use understanding particular surface information of the patient life systems as an advanced info, control the surface information to an ideal frame utilizing CAD programming, handling likewise for current looking and check quality and afterward preparing in fast prototyping machine for creation. The article portrays the use of fast prototyping (RP), 3D examining and programming instruments for the orthosis configuration process. [59]

Now in below papers we will try to focus on software forward engineering process

4.16)Data Mining for Software Engineering Process

Twitter has made a remarkable opportunity for programming engineers to screen the feelings of substantial populations of end-clients of their product. To defeat these difficulties, this extended unique presents a three-overlay strategy that is aimed at utilizing Twitter as a primary wellspring of specialized feedback that programming designers can profit by. Our investigation is led utilizing a dataset of tweets gathered from the Twitter bolsters of three programming frameworks. **[60]**

4.17)Tool for requirements formalizing SEP

This paper depicts a device confirm be utilized to formalize the product necessities. The instrument centers the necessities and proselytes it to Z-determination naturally. Experiments are produced to express the right and fulfillment of the necessities utilizing the FASTEST device for the determined z determinations. **[61]**

4.18)CSEPM.Continues SEP MetaModel

Programming engineers need to adapt to vulnerabilities and evolving prerequisites. The capacities to keep up high code quality through audits, to routinely discharge programming, and to gather and organize client input, are essential for persistent programming building (CSE). Notwithstanding, there exists no product procedure metamodel that handles the persistent character of programming building. In this paper, we portray an

experimental procedure metamodel for nonstop programming building called CSEPM, which regards improvement exercises as parallel running work processes and permits fitting and customization. **[62]**

4.19)Project management and requirement Engineering

Prerequisites building (RE) and necessities change administration (RCM) both are considered as extremely difficult exercises because of requesting rich interchanges. Since it is important to address land and social contrasts in GSD, this prerequisite makes RE and RCM testing. In the first place, the systems with the marvels of particular task administration are proposed for RE and RCM. By using the examined information, our outcomes demonstrate the critical effects of the two systems (i.e., RE and RCM) in the GSD condition. **[63]**

4.20)Text mining and knowledge generation

Being programming building positions a portion of the such sort of employments, there is a significant hole between work postings and contracting dexterous specialists in numerous product designing associations. In this paper, we will present the model of a web application that aides distinguishing Technical Knowledge (TK) in programming improvement, to fill in as an instrument in the enlisting procedure of programming building positions, and in ability administration. We propose an approach to utilize NLP and TM to distinguish learning profiles for Software Engineering Positions.**[64]**

4.21) Systems and software engineering,measurement process.

The estimation procedure is relevant to framework and programming building and administration disciplines. The procedure is portrayed through a model that characterizes the exercises of the estimation procedure that are required to enough indicate what estimation data is required, how the measures and investigation comes about are to be connected, and how to decide whether the examination comes about are legitimate. It distinguishes the exercises and undertakings that are important to effectively recognize, characterize, select, apply, and enhance estimation inside a general task or hierarchical estimation structure. [65]

4.22)Software Engineering Life Cycle Processes - Risk Management.

Because of balloting in ISO, the mission statement was adjusted as takes after: "The motivation behind this standard is to give providers, acquirers, designers, and administrators with a solitary arrangement of process prerequisites appropriate for the administration of a wide assortment of dangers. This standard does not give nitty gritty hazard administration procedures, but rather centers around characterizing a procedure for chance administration in which any of a few methods might be connected. [66]

4.23) Big Data Analysis for software engineering application process

Growing extensive scale programming ventures includes enormous endeavors at each phase of the product improvement life cycle (SDLC). This drove scientists and professionals to create programming procedures and techniques that will help programming designers and enhance their activities. Thinking about the unmistakable qualities of enormous information and the accessible foundations, apparatuses and improvement models, we have to make an orderly way to deal with the SDLC exercises for BDAA advancement. It is important to precisely look at this space and embrace the product forms that best serve the engineers and is sufficiently adaptable to address the diverse qualities of such applications**.** [67]

4.24)Global Software product engineering approach

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. This paper shares the experience from such an activity embraced in an Internet based item organization doing programming item designing crosswise over India and USA**.[68]**

4.25) Model for RE in Big Data

Most predominant programming building philosophies accept that product frameworks are created without any preparation to catch business information and in this manner produce reports. These qualities could be conceivably released and increased from the bits of knowledge found by information researchers through information mining process. Information mining may include overlaying and consolidating information from various sources to separate information designs. In this paper, we display another prerequisite building model that permits programming designers and information researchers to find these qualities as one as a feature of programming necessity process. We likewise show how the proposed prerequisite model catches and communicates business esteems that released through huge information investigation utilizing an adjusted utilize case chart.[69]

4.26)Process Theories and Taxonomies in Software Engineering

Programming building is progressively worried about hypothesis because the foundational learning including speculations gives an essential contrast to the viable information communicated through strategies and methods. Luckily, much direction is accessible for producing and assessing speculations for clarifying why things happen (change hypotheses). Shockingly, little direction is accessible concerning speculations for clarifying how things happen (process hypotheses), or speculations for examining and understanding circumstances (scientific categorizations). This paper accordingly endeavors to clear up the nature and elements of process speculations and scientific categorizations in programming building research, and to blend methodological rules for their age and assessment**.** [70]

5) Research Question focused area is

How Software reengineering leads to new product with quality assurance attributes?

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?

Ans: 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. [71]

How Software Reengineering leads to quality assurance?

Ans: As in fig1.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. After re-engineering quality product chances increases but few percent chances can say lack of quality. After reengineering for the assurance of quality, we test quality parameters in developed product.

Here we will try to elaborate the concept of QA w.r.t re-engineering with proposed theories.

5.1) Quality Measurement an object oriented Approach

There are numerous question arranged programming quality estimation systems. This paper uses CK measurements to quantify characteristics per form of an open source programming in particular Statcato. This paper presumes that the nature of programming has enhanced amid its lifecycle, despite the fact that its highlights increment.[72]

5.2) Requirement Specification to quality

The disappointment and accomplishment of any product fundamentally relies upon a specialized archive known as Software Requirement Specification (SRS) report, as it contains all prerequisites and highlights of the item. Parsing Requirement (PR), Requirement Mapping utilizing Matrix (RMM), Addition of Requirements in SRS format and Third Party Inspection. Prerequisite Engineering Process will give the expected contributions to PR after the usage of its cosmology rules finish of necessities will be accomplished. An outsider assessment will be directed to check the prerequisites of the customer and SRS. In the wake of reviewing SRS utilizing assessment models and appointing Total Quality Score (TQS) outsider will present a point by point answer to group of Requirement Engineers (RE).[73]

5.3) Design patterns impact on software quality

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. We additionally make a stride once again from these examinations and reevaluate the part that outline examples should play in programming advancement. At last, we plot a few roads for future research take a shot at configuration designs, e.g., the recognizable proof of the examples extremely utilized by engineers, the speculations clarifying the effect of examples, or their utilization to raise the deliberation level of programming dialects.[74]

5.4) Software Quality analysis in practice

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. I will cover hard lessons learned on the best way to have an effect in certifiable undertakings, amazing aftereffects of apparently inconsequential methodologies, the part of programming perceptions in advertising and our key learnings in exchanging research from the scholarly world to rehearse.[75]

5.5) Methodology for SQ improvement

Quality is the most essential factor for software improvement as it for the most part characterizes consumer loyalty that is specifically identified with the achievement of a software venture. 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.[76]

5.6) 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. The as a matter of first importance objective is to assess the outline designs regarding their plan and quality traits. The second target is to give an outline as how these plan designs influence the software quality.[77]

5.7)Deploying Software Analytics in Multinational Organization

Actualizing software designing investigation arrangement postures difficulties and offers critical incentive for the comprehensively conveyed software improvement association at ABB. Since software advancement exercises in nimble systems rotate around the group, ABB chose to execute an examination arrangement concentrated on group measurements as a feature of its Software Development Improvement Program. Utilizing key pointers centered on group change, scientists found that groups could deal with their exercises with measurements, for example, process duration.[78]

5.8) Object oriented code refactoring and quality

Software refactoring is a support undertaking that delivers code rebuilding to enhance its quality. This examination exhibits a methodical writing survey that totals, condenses, and talks about the consequences of 76 important essential investigations (PSs) concerning the effect of refactoring on a few inner and outside quality characteristics. We dissected the PSs in light of an arrangement of order criteria, including software quality characteristics and measures, refactoring situations, assessment approaches, datasets, and affect comes about. The outcomes demonstrated that distinctive refactoring situations here and there implicitly affect diverse quality traits. [79]

5.9) Improve from open source SPM

Software designing administration inquire about has been directed for a long time. Scientists have confronted a heap of difficulties in acquiring dependable information and related measurements, creating down to earth assessing models, and affecting undertaking changes inside industry. This paper investigates how huge information from OSS can enhance software designing and administration hones. [80]

5.10) Software repository mining code review

We display Digits, an apparatus to naturally produce code survey remarks, offering plan direction on forthcoming changes, in view of bits of knowledge picked up from mining chronicled changes in source code archives. We center around the engineer understanding, the requirements that must be met in adjusting scholarly research to deliver a device that was valuable to designers, and the viability of the outcomes by and by.[81]

5.11) Real-time code quality assessment

Code measurements can be utilized to survey the inner nature of software frameworks, and specifically their adherence to great plan standards. For sure, they take a code part as information and evaluate a quality trait (e.g., code coherence) by giving a number as yield. We display RETICULA (RealTime Code qualityAssessment), a module for the IntelliJ IDE to help engineers in seeing code quality amid software advancement. With the imagined comes about, designers can pick up experiences about the nature of their code.[82]

5.12) Empirical Validated Quality Model

In this paper we give a meaning of code quality for Puppet code and a robotized strategy for estimating and rating Puppet code quality. To this end, we initially investigate the thought of code quality as it applies to Puppet code by playing out a study among Puppet engineers. To touch base at this estimation display, we get fitting quality measurements from our overview come about and from existing software quality models. We approve our meaning of Puppet code quality and the estimation display by an organized meeting with Puppet specialists and by contrasting the device results and quality judgments of those specialists.[83]

5.13) Code clone with Bugs and quality of code

Code clone is a monstrously examined code smell. This paper introduces a similar report on the attributes of surrey and non-carriage clones from a code quality point of view. In the light of 29 code quality measurements, we ponder carriage and non-surrey clones in 2,077 corrections of three software frameworks written in Java.[84]

5.14) Software refactoring method level clustering network

In this examination, we portray a framework level various refactoring calculation, which can distinguish the move technique, move field, and concentrate class refactoring openings consequently as indicated by the rule of "high attachment and low coupling." The calculation works by blending and part related classes to acquire the ideal usefulness conveyance from the framework level. In view of correlations with related research and surveying the refactoring comes about utilizing quality measurements and observational assessment, we demonstrate that the proposed approach performs well in various frameworks and is gainful from the point of view of the first engineers. [85]

Table1. Re-engineering Major areas

[2] Sr#	[3] **Title**	[4] **Authors**	[5] **Issues Found**	[6] **Methodology** [7] **Applied**	[8] **Major Input**	[9] **Major Output/Finding**	[10] **Ref#**
[11] 1.1	[12] Human beings and Machine Interfacing	[13] Bernhard Dorninger,Michael Moser.(2018)	[14] Advances in technology, cost pressure, change in requirements	[15] 1)Statistical analysis used for requirements evaluation [16]	[17] 1)Human needs [18] 2)Technology changes [19] 3)Machine interfacing	[20] Reengineered machined, configured and maintainable with advancements of technology.	[21] [22] [2]
[23] 2.1	[24] Code smells findings and refactoring	[25] Ghulam Rasool,Zeeshan Arshad.(2016)	[26] Code smell leads to poor quality of software	[27] 22 different types of code smell detection techniques used	[28] 8 different Open source software	[29] Founded results detected all code smells and proposed solutions	[30] [3][4]
[31] 3.1	[32] Business Process Reengineering (BPRE)	[33] SaeedehGhanadbashi , Raman Ramsin.(2016)	[34] Old business process leads to business failure	[35] Framework with reengineering process add up adopted	[36] Situational process	[37] Reengineered business framework leads to success	[38] [5]
[39] 4.1	[40] Improving re-engineering performance	[41] A. CathreenGraciamary , Dr. M.Chidambaram.(2018)	[42] Re-engineering problems, old system usability	[43] Data converted into bytes	[44] Bytes and Datasets	[45] Access time of reengineered system is very much lower as compared to old, then performance increases.	[46] [6]
[47] 5.1	[48] Safe and secure process in 21th century	[49] K.R. Wallace.(2014)	[50] Requirements integrity in legacy system is a risk	[51] Template implemented for integrity and risk analysis	[52] More requirement on customer demand	[53] This approach provide solution for risk factors to manage integrity and consistency of system	[54] [7]
[55] 6.1	[56] Re-engineering of data storage	[57] P. Hunter.(2013)	[58] Data storage for big data becoming a problem	[59] Data set pattern matching and massive storage can be resolved	[60] Hard disk drive and solid state drive	[61] Data intensive applications can be managed by cloud computing	[62] [8]
[63] 7.1	[64] Raising software engineering as a search problem	[65] J. Clarke , J.J. Dolado , M. Harman , R. Hierons(2003)	[66] How to test data generation, module clustering and cost/effort prediction.	[67] Met heuristic techniques	[68] Implementation of technique in SE areas [69]	[70] Met heuristic search tried to raise software engineering as a huge search problem [71]	[72] [9]
[73] 8.1	[74] Software components mismatch detection and resolution	[75] Egyed , N. Medvidovic. [76] (2000). [77] [78]	[79] Identification of software components miss-match	[80] 1)Model-based development (e.g. architectural modeling)	[81] software modeling and analysis techniques, all tool work as input for resolving risks	[82] Many correct components matching problem has been resolved	[83] [10]

[84] 9.1	[85] Managing wireless networks spectrum and reengineering	[86] K. Rafique (2011) [87] [88] [89] [90] [91] [92]	[93] How we can utilize spectrum and challenges in address formation	[94] Spectrum framework in managing addresses	[95] Technologies involved in promoting flexible licensing approaches	[96] It provide us suggestions for a regulatory framework accommodating market approaches as well as an effective control mechanism to resolve the conflicts and complexities of the future market.	[97] [11] 468
[98] 10.1	[99] Object oriented Based legacy systems [100]	[101] AmanJatain.(2015)	[102] Old obect oriented legacy systems are not supportable for new technologies	[103] Tried to convert into modular systems	[104] Old legacy systems with object oriented	[105] Software with components or modules work better with advance technologies.	[106] [12]

[107]
[108]
[109] Table2. Re-Structuring Process Area
[110]

[111] Sr #	[112] **Title**	[113] **Authors**	[114] **Issues Found**	[115] **Methodology** [116] **Applied**	[117] **Major Input**	[118] **Major Output/Finding**	[119] **Ref #**
[120] 2.1	[121] Signal processing Method for restructure	[122] CharilaosPetrou.(2016)	[123] Signal processing with big data	[124] principal component analysis(PCA)	[125] Big data from different organizations [126]	[127] Big data can be processes by different signal processing techniques.	[128] [129] [14]
[130] 2.2	[131] On chip memory restructure for bandwidth	[132] Jason Cong,PengWei,CodyHao Yu.(2107)	[133] High level synthesis fail to put arrays design on-chip	[134] Automated on-chip buffer restructuring can resolve the issue	[135] Inserting High level designs on-memory chip buffers	[136] Our automated source-to-source code transformation tool improves the performance of a broad class of high level synthesis	[137] [15]
[138] 2.3	[139] Effect on Information technology field	[140] Fernando Szimanski,Anivaldo S. Vale.(2018)	[141] IT have to fulfill customer requirements	[142] Business models and analyzing tools	[143] Changing internal processes customization	[144] IT provides services quality, customer satisfaction and engagement, as well as transparency on IT projects. [145]	[146] [16]
[147] 2.4	[148] Layouts and visual learning Models	[149] KashyapTodi,Jussi Jokinen.(2017)	[150] Computational problems exists in software due to layout and designs	[151] Human System Visualization	[152] Users software layouts and interfaces	[153] Well organized software systems with user friendly interfaces are developed	[154] [17]

[155]12.5	[156]Restructuring object-oriented software	[157]Amit Rather. [158](2017)	[159]Due to increase in features and advance capabilities ,we restructure software	[160]Cohesion techniques are used to rebuilt the classes of OOP based system	[161]Existing software systems	[162]Welldesigned,organized,restructured,new systems are obtained after restructuring	[163][18]
[164]2.6	[165]Automated refactoring approach to remove code complexities	[166]Nathan ManeraMagalhães. [167](2017)	[168]Unnecessary structural complexity may occur, in which a program has a cycloramic complexity	[169]Identify complexity, restructuring code and finding flow graphs	[170]Software source code, software applications	[171]The approach is able to support unnecessary cycloramic complexity remove	[172][19]
[173]2.7	[174]Replacing code type with state and subclasses	[175]JyothiVedurada. [176](2017)	[177]Large volume of code causes complexities	[178]Replace code with sub classes	[179]Real time JAVA applications	[180]Restructured well designed and user friendly system is developed	[181][20]
[182]2.8	[183]Issues in quality of code	[184]HiekeKeuning [185](2017)	[186]Code quality is a big issue in these days	[187]Functions clearity,expressions finding demoularization	[188]Wrong coded programs by students	[189]Clear and refactored with clear quality programs are obtained	[190][21]
[191]2.9	[192]Scientific apps via code refactoring	[193]Ana Rodriguez. [194](2017)	[195]Mobile rely on batteries and APPS use much battery	[196]Code refactoring reduce power consumption	[197]Mobile apps with old interfaces	[198]Via code restructuring new coded apps reduced the power usage	[199][22]
[200].[201]10	[202]Automated code refactoring and reuse	[203]Didier Rémy.(2017)	[204]Inductive data types and parametric polymorphism are two common problems	[205]Inductive data-types and parametric polymorphism	[206]Old systems with presence of code smells	[207]By adding or dropping some parts of codes then it is possible to make automated pointing system that can be helpful in	[208][23]
[209].[210]11	[211]Relationship between refactoring and change	[212]Rocco Oliveto.(2017)	[213]New features introduction in any product lead toward refactoring	[214]Fault Repairing Modification, Feature Introduction Modification, and General Maintenance Modification	[215]Programs with inconsistent and pattern mismatch	[216]New features are introduced with advance capabilities	[217][24]
[218].[219]12	[220]Impact of refactoring on code smell	[221]Diego Cedrim.(2017)	[222]Smells in a program represent indications of structural quality problems	[223]Analyzing 23 software systems,%ages of refactoring factors obtained	[224]23 different software systems	[225]Some time Refactoring leads to more smells in code,instead of removing existing once	[226][25]
[227].[228]13	[229]Software restructuring model driven	[230]Dennis Dams. [231](2018)	[232]Software available for restructuring need to be customized	[233]COM interfaces analysis from High-level code model to low-level	[234]Interfaces of a large industrial software component	[235]Method of restructuring obtained called model driven	[236][26]
[237].[238]14	[239]Restructuring tool for java software	[240]Danilo Santos.(2016)	[241]Software need to be restructured	[242]Instead of classes,pakages are proposed	[243]JAVA Software old versions	[244]High coupling and less cohesion products developed	[245][27]

[246]
[247] Table3. Legacy software systems and reengineering
[248]

[249] Sr #	[250] **Title**	[251] **Authors**	[252] **Issues Found**	[253] **Methodology** [254] **Applied**	[255] **Major Input**	[256] **Major Output/Finding**	[257] **Ref #**
[258] 3.1	[259] Legacy software systems reengineering ?	[260] Stromasys.(2016)	[261] Old systems are not capable to adopt advance technology	[262] Re-engineering techniques to add more features	[263] Old legacy systems hadwares/softwares	[264] New products obtained with advance features	[265] [29]
[266] 3.2	[267] Automatic components reengineering in legacy systems	[268] James.J. Mulcahy.(2017)	[269] When many components of legacy systems ,to develop a system cause complexities	[270] adaptive autonomic interaction managers (AIMs)	[271] Legacy software of many components based systems	[272] Reengineered product with addition of several old systems obtained	[273] [30]
[274] 3.3	[275] Managerial dimensions on reengineering process [276]	[277] Anand Rajavat. [278] (2014)	[279] 1)Current market strategy, 2)product competition, 3)operational constraints	[280] Analyzing existing software systems to develop methodology	[281] Identification and development of various metrics and important measures to measure effect of individual risk component	[282] Risk factors like cost and time for reengineering process improved	[283] [31]
[284] 3.4	[285] Software reengineering supported by database reverse engineering w.r.t legacy systems	[286] Stefan Strobl. [287] (2009)	[288] completely migrating the IS to an up-to-date and homogeneous platform	[289] Database reverse engineering of the system	[290] Tables and columns of legacy databases	[291] DRE process resulted in the development of a high-level categorization of the data model	[292] [32]
[293] 3.5	[294] Legacy systems reengineering to metaprogramming	[295] Ankit B. Desai.(2106)	[296] MDD is subjective area and need to redesign the architecture	[297] Architecture, process,system reengineering	[298] Old legacy software systems models of architectures	[299] New developed systems with advance architectures and designs	[300] [33]
[301] 3.6	[302] Refactoring Cost Estimation (RCE) Model	[303] Paulo Eduardo Papotti. [304] (2012)	[305] Cost effects on refactoring	[306] A model is proposed - RCE	[307] Old legacy system with effective cost unknown	[308] Model driven approach proposed to find cost	[309] [34]

ISSN 2250-3153

[310]3.7	[311]Software architecture for power market	[312]Qing Zhao.(2003)	[313]electronic commerce solution needed for deregulated power markets,	[314]A heterogeneous software architectural style is designed for the Web-based software	[315]Heterogeneous software with legacy systems architecture	[316]A wrapping technique is designed to transfer the legacy energy-management system (EMS) modules into reusable components	[317][35]
[318]3.8	[319]Reengineering risk quantify for legacy system	[320]AnandRajavat.(2012)	[321]Risk in legacy systems performance	[322]Comprehensive model for quality assurance	[323]Legacy systems with old patterns	[324]Rish management with advance qualities	[325][36]
[326]3.9	[327]Ways of evaluation in legacy systems	[328]Sayed Muqtada Hussain; ShahidNazir Bhatti.(2017)	[329]Old systems are called legacy systems	[330]Complicated problems removed	[331]Organizational old software systems	[332]Reengineered systems with advance working	[333][37]
[334]3.[335]10	[336]Software reengineering for DSP-based systems on a chip	[337]ZdravkoPanjkov; Mihajlo Katona; [338](2012)	[339]An increased complexity of modern computer based systems	[340]It is accompanied by advanced software architectures embedded	[341]DSP based system on a chip, including system verification	[342]The critical reengineering decisions are elaborated with given results and lessons learned during project implementation.	[343][38]
[344]3.[345]11	[346]Requirements reconciling via reengineering	[347]Jianchu Huang; Hongji Yang; [348](2011)	[349]Software requirements and implementation are not always reconciled	[350]a context-aware requirements elicitation approach to reconcile the gap is used	[351]Open-source location-aware legacy system	[352]Gap between implementation and requirements elicitation removed	[353][39]
[354]3.[355]12	[356]Legacy escaping safety in critical embedded system	[357]Michael Stilkerich; Jens Schedel; [358](2011)	[359]Java have not yet found their way into the domain of deeply embedded systems	[360]an approach that allows to combine legacy software components	[361]Java based legacy systems	[362]The cost of this mixed-mode operation is on the same scale as the pure operation	[363][40]
[364]3.[365]13	[366]Reengineering of enterprise software for cloud computing	[367]Hong Zhou; Hongji Yang; [368](2010)	[369]Enterprise software management is a problem in real time systems	[370]Cloud computing tried to follow up for reengineering	[371]Old legacy enterprise level system	[372]Advance software systems with cloud computing concepts	[373][41]
[374]3.[375]14	[376]IT Infrastructure Reengineering	[377]Luis Ferreira da Silva; Fernando Brito e Abreu. [378](2010)	[379]Poor IT infrastructure leads to issues in software	[380]IT infrastructure improvement and reengineering approaches	[381]Old systems with old system	[382]New advance featured product with latest infrastructure	[383][42]

[384]
[385]
[386] Table4. Software Reverse Engineering Process
[387]

[388]Sr#	[389]**Title**	[390]**Authors**	[391]**Issues Found**	[392]**Methodology** [393]**Applied**	[394]**Major Input**	[395]**Major Output/Finding**	[396]**Ref#**

[397]4.1	[398]Reverse engineering and open source software	[399]Taher Ahmed Ghaleb. [400](2016)	[401]Structures of open source software need to reverse engineer	[402]Static analysis of source code and dynamic analysis of program	[403]Open Source software	[404]For reverse engineering good approach obtained	[405] [406][45]
[407]4.2	[408]Software for IC Reverse Engineering	[409]Gyungtae Kim; Ming Ma. [410](2018)	[411]CAD Tools for IC reverse engineering	[412]Reverse engineering process, Big Image acceleration algorithm	[413]Integrated circuit(IC)	[414]RE with the self-manufactured ROIC chip	[415][46]
[416]4.3	[417]Conventional Software Reverse Engineering	[418]Alexander Bergmayr; Hugo Bruneliere;. [419](2016)	[420]Semantic Web Enabled Software Engineering	[421]Framework to solve semantic web problem	[422]Integration among research semantic web technologies	[423]Implementation of semantic web enable new technologies	[424][47]
[425]4.4	[426]Trace analyzer for real time software	[427]M P S Bhatia; Akshi Kumar;. [428](2016)	[429]The process of creating models of existing system components is often difficult and time consuming	[430]CoreTAna, a novel tool that derives an AUTOSAR compliant model	[431]legacy code has to be re-used or information [432]re-use	[433]CoreTAna's current features and discusses its benefits for reverse engineering	[434][48]
[435]4.5	[436]Process mining event log analysis [437]	[438]Andreas Sailer; Michael Deubzer; [439](2016)	[440]Process mining. Implementations, documentation, and a screen-cast	[441]Event log analysis and process mining concepts	[442]The behavior of a (legacy) software system	[443]Present the State chart Workbench	[444][49]
[445]4.6	[446]Hierarchical analysis recursion modeling [447]	[448]MaikelLeemans. [449](2018)	[450]The work can be positioned in-between reverse engineering and process mining	[451]Recursion aware process model discovery technique	[452]A novel hierarchy and recursion extension to the process tree	[453]Event logs demonstrate the feasibility and usefulness	[454][50]
[455]4.7	[456]Hierarchical analysis recursion modeling	[457]MaikelLeemans [458](2018)	[459]	[460]	[461]	[462]	[463][51]
[464]4.8	[465]Sentimantics programming languages and software support	[466]William Steingartner. [467](2017)	[468]Programming language need to support semantic software systems	[469]Packages modules prepared for programming language	[470]Software and semantic programming language	[471]Extended software packages for semantic programming	[472][52]
[473]4.9	[474]Software clustering hill climbing	[475]MasoudKargar. [476](2017)	[477]Software need to be clustered for quality	[478]Hill climbing mechanism used	[479]Extracting the Call Dependency Graph (CDG) from the source code	[480]Software Clustering helps the software engineer to clustering the large-scale software systems	[481][53]

[482]4. [483]10	[484]Code obfuscation and Reverse Engineering	[485]Asish Kumar Dalai. [486](2017)	[487]Re-engineer the code leads to software piracy	[488]Code obfuscation technique	[489]Dis-assembling the executables to recover the source code/assembly code	[490]Quantify the degree of obfuscation, stealth of the code	[491][54]
[492]4.	[494]Software code smell and	[495]Anshul Rani. [496](2017)	[497]Software versioning ,poor	[498]Efficient refactoring	[499]Low quality	[500]Initial version of project can be used as reference	[501][5

[493]11	multiple versions		design	strategies	software with bad designs	architecture for purpose of reverse Engineering	5]
[502].[503]12	[504] Fuzzy framework for network reverse engineering [505]	[506] BernhardsBlumbergs. [507] (2017)	[508] File and protocol fuzzing	[509] A bit-aware fuzzing framework	[510] Secure software development life-cycle	[511] A systematic approach is proposed and tool prototype developed for the cyber red teaming purposes	[512][56]
[513].[514]13	[515] Reverse engineering awareness [516]	[517] Ana Maria da Mota Moura. [518] (2017)	[519] Software changes, it may become more difficult to under-stand, to be changed and harder to be reusable	[520] Requirements as soft-goal in a goal-oriented model	[521] Changes in user requirement and old legacy software	[522] Goal model provide a high-level description of both the adaptation logic as well as the application logic	[523][57]
[524].[525]14	[526] Using abstract syntax tree Source code to sequence diagram	[527] EsaFauzi; BayuHendradjaya; [528] (2016)	[529] From source code to reverse engineering process was need	[530] AST(Abstract syntax tree)that convert source code to structure	[531] Source code from the software based systems	[532] Sequence diagram structure and reverse engineering throughput	[533][58]
[534].[535]15	[536] Using reverse engineering and rapid prototyping for patient	[537] RafalKudelski; Piotr Dudek; [538] (2017)	[539] Reverse engineering methods are combined with rapid prototyping to create patient-specific orthoses	[540] 3D digitizing, reverse engineering and polygonal-surface software, SLS RP and 3D printing.	[541] Reverse engineering methods are combined with rapid prototyping to create patient-specific orthoses	[542] The article describes the utilization of rapid prototyping (RP), 3D scanning and software tools for the orthosis design process	[543][59]

[544]
[545]
[546] Table5. Software Forward Engineering Process
[547]

[548] Sr#	[549] **Title**	[550] **Authors**	[551] **Issues Found**	[552] **Methodology** [553] **Applied**	[554] **Major Input**	[555] **Major Output/Finding**	[556] **Ref#**
[557].[558]16	[559] Data Mining for Software Engineering Process	[560] Grant Williams; Anas Mahmoud; [561] (2017)	[562] Data Mining for software engineering	[563] Data Mining and software development approaches	[564] Data of old software systems	[565] Normalized data obtained for engineering new software	[566] [567][60]
[568]4.	[570] Tool for requirements	[571] V. Madhan; V. K. G.	[572] Software Engineering	[573] Software Engineering	[574] Extracting requiremen	[575] Well organized SEP approach and software	[576][6

[569]17	formalizing SEP	Kalaiselvi; Donald J. P;(2017)	process need a Tool	process and latest tools	t from customer	systems	1]
[577]4.[578]18	[579]CSEPM.Continues SEP Meta Model	[580]Stephan Krusche; Bernd Bruegge.(2017)	[581]SEP need improvement for quality	[582]Model driven approach for development	[583]Software systems and model	[584]SEP model from meta models approaches	[585][62]
[586]4.[587]19	[588]Project management and requirement Engineering	[589]Muhammad Shafiq; Qinghua Zhang.(2017)	[590]Unbalanced requirements have invalid quality points	[591]Project management methodologies	[592]Large organizational projects [593]	[594]Projects handling from old systems requirements	[595][63]
[596]4.[597]20	[598]Text mining and knowledge generation	[599]Rogelio Valdez-Almada; Oscar M. Rodriguez-Elias.(2017)	[600]How text mining generate knowledge	[601]Data Mining and extracting knowledge	[602]Data gathered from different systems	[603]From dta getting knowledge for software systems	[604][64]
[605]4.[606]21	[607]Systems and software engineering, measurement process. [608]	[609]ISO/IEC/IEEE 15939.(2017)	[610]Engineering measurements are not possible without SEP	[611]Engineering and control management approaches	[612]Software product and customer needs	[613]An organized process obtained for SEP management	[614][65]
[615]4.[616]22	[617]Software Engineering Life Cycle Processes - Risk Management.	[618]IEEE P16085/CD, [619](2018)	[620]Software development risk is big issue in large projects	[621]Risk factor management and Software process cycle	[622]From customer requirement following a process	[623]Managing risks of cost, possibility and availability of product	[624][66]
[625]4.[626]23	[627]Big Data Analysis for software engineering application process	[628]Jameela Al-Jaroodi; Brandon Hollein. [629](2017)	[630]SEP and big data analysis led to complexities	[631]Analytical tools for big data management	[632]Gathering data from different software systems	[633]Adopting softare process with proposed throughput analysis	[634][67]
[635]4.[636]24	[637]Global Software product engineering approach	[638]Bhaskarjyoti Das; [639](2017)	[640]Software engineering approaches and global impact	[641]Software Engineering methods and worldwide methods	[642]Input of customer data	[643]Engineering product with developed features	[644][68]

[645]4.[646]25	[647]Model for RE in Big Data	[648]Hamza Hussein Altarturi; Keng-Yap Ng. [649](2017) [650]	[651]Requirements engineering process need to follow valid steps	[652]Requirement engineering approaches and Big data process model	[653]Requirements engineering from software	[654]Validity tests of requirements gathered from big data	[655][69]
[656]4.[657]26	[658]Process Theories and Taxonomies in Software Engineering	[659]Paul Ralph; [660](2018)	[661]Theories effects on taxonomies of software	[662]Process management prom the different theories	[663]Software process steps	[664]Taxonomies based developed Environment	[665][70]

[666]
[667]
[668] Table6. Software Quality Assurance(SQA) Process
[669]

[670] Sr #	[671] **Title**	[672] **Authors**	[673] **Issues Found**	[674] **Methodology** [675] **Applied**	[676] **Major Input**	[677] **Major Output/Finding**	[678] **Ref #**
[679] 5.1	[680] Quality Measurement an object oriented Approach	[681] IwanBinanto; Harco Leslie Hendric Spits Warnars. [682] (2018)	[683] Quality management is a problem	[684] CK Matrices	[685] Old versions of software systems	[686] object-oriented software quality measurement techniques	[687] [688] [72]
[689] 5.2	[690] Requirement Specification to quality	[691] Syed Waqas Ali; QaziArbab Ahmed.(2018)	[692] SRS did not fulfill the quality requirements	[693] Parsing Requirements(PR),Mapping Req [694]	[695] SRS Based systems	[696] New product with Total quality management(TQM) [697] With advance methods	[698] [72]
[699] 5.3	[700] Design patterns impact on software quality	[701] FoutseKhomh; Yann-GaëlGuéhéneuc. [702] (2018)	[703] Software proper design is an issue for development	[704] Pttern matching and modular techniques	[705] Old patterns software systems	[706] Impact of patterns on high level programming from base to end	[707] [74]
[708] 5.4	[709] Software Quality analysis in practice	[710] ElmarJuergens. [711] (2018)	[712] Problematic code quality leads to failure	[713] Quality analyses built upon this community's research	[714] Code smells in software used as input	[715] With the better in code quality with reengineering mechanism can lead to quality attribute	[716] [75]
[717] 5.5	[718] Methodology for SQ improvement	[719] Muhammad Azeem Akbar; Jun Sang.(2018)	[720] Customer satisfaction is a big issue	[721] AZ-Model driven approach is used	[722] Already existing software cycle	[723] New software cycle proposed with advance features	[724] [76]
[725] 5.6	[726] Design patterns and Quality Assurances	[727] Muhammad Noman Riaz. [728] (2018)	[729] Invalid approaches for designing mechanisms	[730] Validity tests of design patterns w.r.t Quality	[731] Previous pattern software	[732] Four main quality attributes involved in quality assurence	[733] [77]

[734] 5.7	[735] Deploying Software Analytics in Multinational Organization	[736] Vinay Augustine; John Hudepohl. [737] (2018)	[738] Software deployment is a problem in these days	[739] Deployment methodology with validity tests	[740] Software systems with deployed systems	[741] Multinational organizations need to adopt validity approaches to deploy software	[742] [78]
[743] 5.8	[744] Object oriented code refactoring and quality	[745] Jehad Al Dallal; Anas Abdin. [746] (2018)	[747] Internal and external attributes need to be improved	[748] Code restructuring and refactoring scenario	[749] 76 relevant studies discussed	[750] Sometime refactoring scenarios leave some draw backs on quality	[751] [79]
[752] 5.9	[753] Improve from open source SPM [754]	[755] John Jagtiani; Christian Bach. [756] (2018)	[757] Open source software need to be well organized	[758] Software project management techniques	[759] Open source software systems	[760] It is expected that these tangible research outcomes would be of keen interest to practitioners	[761] [80]
[762] 5. [763] 1	[764] Software repository mining code	[766] Robert Chatley; Lawrence Jones. [767] (2018)	[768] Code need to review for the purpose of	[769] Mining historical changes and repositories	[770] Code from old systems present for	[771] Code review comments, design guidance	[772] [81]

0	review [765]		quality	dependences	review		]
[773].[774]11	[775] Real-time code quality assessment	[776] Luigi Franzio; Bin Lin; Michele Lanza. [777](2018)	[778] Real time code based software with quality dependencies	[779] Code readability methodology	[780] Code components as input access to code quality	[781] Visualized results with the validity of code contains and readability	[782][82]
[783].[784]12	[785] Empirical Validated Quality Model	[786] Eduard van der Bent; Jurriaan Hage. [787](2018)	[788] Software dependencies on puppet software code qualities	[789] Puppet code quality and the measurement model by a structured interview with Puppet experts	[790] Numerous code based software systems	[791] The validation shows that the measurement model and tool provide quality judgments of Puppet code	[792][83]
[793].[794]13	[795] Code clone with Bugs and quality of code	[796] Md Rakibul Islam; Minhaz F. Zibran. [797](2018)	[798] Cloning and bugs issues are present in code quality	[799] Characteristics of buggy and non-buggy clones from a code quality perspective	[800] Code clone is an immensely studied code smell	[801] The findings from this work add to the characterization of buggy clones	[802][84]
[803].14	[804] Software refactoring method level clustering network [805]	[806] Ying Wang; Hai Yu; Zhiliang Zhu. [807](2018)	[808] Software refactoring methods need to improve	[809] Non-inheritance and inheritance hierarchies without changing the code behaviors	[810] Using a series of preprocessing steps and preconditions, the "bad smells"	[811] An open source tool is implemented to support the proposed approach	[812][85]

V. Conclusion

This paper gives view of researches for the various areas of software reengineering. This paper spread light on the dimensions of i.e. 'Refactoring', 'Reverse engineering', 'Forward engineering', 'Restructuring' with different types of restructuring(code,patterns,designs,architecture,documents,coding,testing,developing).Forward engineering moved with this sequence(Data restructuring, requirements elicitation,designing,implementation,deployment,user needs testing) and reverse engineering with sequence(From software,implementation,designing,requirement gathering and data restructuring).As we move toward reverse engineering and add some more attributes in data restructuring stage and develop new product with adding more features .Over all purpose of this study is based on empirical searches of other author. For the explanation of concept here, we bring in discussion more than 85 theories of different authors related to different fields. We conclude that if we follow valid procedure to reengineer old legacy software systems without disturbing its external behavior we can set up code quality, designing architectural quality, we can fulfill the customer needs, and we can offer advanced new product with more quality attributes. In future work quality texts behaviors on software system after reengineering can be performed.

References

[1] 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

[2] BernhardDorninger," *Reengineering an industrial HMI: Approach, objectives, and challenges*" IEEE 25th International Conference 20-23.March2018

[3] 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|

[4] 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

[5] SaeedehGhanadbashiand RamanRamsin," *Towards a method engineering approach for business process reengineering*" Volume 10, Issue 2, April 2016, p. 27 – 44

[6] 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

[7] 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

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

[9] 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

[10] A. Egyed, N. Medvidovic." Component-based perspective on software mismatches detection and resolution" Volume 147, Issue 6, December 2000, p. 225 – 236

[11] 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

[12] AmanJatain and Deepti Gaur," *Reengineering Techniques for Object Oriented Legacy Systems*" International Journal of Software Engineering and Its Applications Vol. 9, No. 1 (2015), pp. 35-52

[13] 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).

[14] CharilaosPetrou," *Signal Processing Techniques Restructure the Big Data Era*" Article No. 52, Patras, Greece — November 10 - 12, 2016

[15] Jason Cong, Peng Wei, CodyHao Yu" *Bandwidth Optimization through On-Chip Memory Restructuring for HLS*" Austin, TX, USA — June 18 - 22, 2017

[16] 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

[17] KashyapTodi, Jussi Jokinen
" *Familiarization: Restructuring Layouts with Visual Learning Models*", 23rd International Conference on Intelligent User Interfaces, Pages 547-558,

[18] 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,

[19] 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

[20] JyothiVedurada," *Refactoring opportunities for replacing type code with state and subclass*", ICSE-C '17, Pages 305-307, Buenos Aires, Argentina — May 20 - 28, 2017

[21] HiekeKeuning," *Code Quality Issues in Student Programs*", ITiCSE '17,Pages 110-115, Bologna, Italy — July 03 - 05, 2017

[22] 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

[23] Didier Remy,Inria, France," *More automated code refactorization and code reuse (invited talk)*", Haskell 2017, Pages 1-1, Oxford, UK — September 07 - 08, 2017

[24] Rocco Oliveto," *An exploratory study on the relationship between changes and refactoring*", ICPC '17 Proceedings of the 25th International Conference on Program Comprehension, Pages 176-185

[25] Diego Cedrim, Alessandro Garcia," *Understanding the impact of refactoring on smells: a longitudinal study of 23 software*", ESEC/FSE 2017, Pages 465-475

[26] Dennis Dams; "*Model-based software restructuring: Lessons from cleaning up COM interfaces in industrial legacy code*". IEEE 25th International Conference, Evolution and Reengineering (SANER), 2018,pages 552 - 556

[27] Danilo Santos; "*SRT — A computational tool for restructuring Java software*", 35th International Conference of the Chilean Computer Science Society (SCCC), 2016, 1 – 12, IEEE Conferences

[28] The diagram is obtained (26 April 26, 2018). https://www.tutorialspoint.com/software_engineering/software_maintenance_overview.htm

[29] The data visited (29 April 2018) https://www.stromasys.com/2016/07/an-overview-of-legacy-software-and-legacy-systems/

[30] 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

[31] AnandRajavat," *Effect of managerial dimensions on reengineering process of legacy software systems*".

2014 Conference on IT in Business, Industry and Government (CSIBIG), Pages: 1 - 6, IEEE Conferences

[32] Stefan Strobl," *Digging deep: Software reengineering supported by database reverse engineering of a system with 30+ years of legacy*", 2009 IEEE International Conference on Software Maintenance, Pages: 407 – 410, IEEE Conferences

[33] Ankit B. Desai," *Refactoring Cost Estimation (RCE) Model for Object Oriented System*", 2016 IEEE 6th International Conference on Advanced Computing (IACC), Pages: 214 – 218, IEEE Conferences

[34] Paulo Eduardo Papotti," *An approach to support legacy systems reengineering to MDD using metaprogramming*",2012 XXXVIII Conferencia Latin-American En Informatics (CLEI),Pages: 1 – 10, IEEE Conferences

[35] Qing Zhao," *A software architecture for power market supporting system and reengineering of legacy EMS*", IEEE Transactions on Power Systems, 2003, Volume: 18, Issue: 1, Pages: 191 – 197, IEEE Journals & Magazines

[36] AnandRajavat," *Decision driven risk measurement model to quantify reengineering risk in stakeholder perspective of legacy system*", 2012 Ninth International Conference on Wireless and Optical Communications Networks (WOCN), Pages: 1 – 5, IEEE Conferences

[37] Sayed Muqtada Hussain," *Legacy system and ways of its evolution*", 2017 International Conference on Communication Technologies (ComTech), Pages: 56 – 59, IEEE Conferences

[38] ZdravkoPanjkov," *A Case Study in Software Reengineering for a DSP-based System on a Chip: Adaptation of Dolby Virtual Speaker*", 2012 IEEE 19th International Conference and Workshops on Engineering of Computer-Based Systems, Pages: 303 – 307, IEEE Conferences

[39] Jianchu Huang," *Reconciling Requirements and Implementation via Reengineering for Context-Aware Service Evolution*", 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops, Pages: 464 – 469, IEEE Conferences

[40] Michael Stilkerich," *Escaping the Bonds of the Legacy: Step-Wise Migration to a Type-Safe Language in Safety-Critical Embedded Systems*", 2011, Pages: 163 – 170, IEEE Conferences

[41] Hong Zhou, "*An Ontology-Based Approach to Reengineering Enterprise Software for Cloud Computing*", Conference, Pages: 383 – 388, IEEE Conferences

[42] Luis Ferreira da Silva, "*Reengineering IT Infrastructures: A Method for Topology Discovery*", 2010 Seventh International Conference on the Quality of Information and Communications Technology, Pages: 331 – 336, IEEE Conferences

[43] NileshJadav,"*How To Reverse Engineer Using Advanced Apk Tool*", Mar 08 2017, https://www.c-sharpcorner.com/article/how-to-reverse-engineer-using-advanced-apk-tool/

[44] Parminder Singh, "*Software reverse engineering*",Student at Punjabi University, https://www.slideshare.net/parrychahal50/software-reverse-engineering-69635162, Published on Nov 29, 2016,

[45] Taher Ahmed Ghaleb, "*The role of open source software in program analysis for reverseengineering*", 2nd International Conference on Open Source Software Computing (OSSCOM), 2016,Pages: 1 - 6,IEEE Conferences

[46] 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

[47] Alexander Bergmayr; Hugo Bruneliere; Jordi Cabot; Jokin Garcia; Tanja Mayerhofer; Manuel Wimmer, "*fREX: fUML-based Reverse Engineering of Executable Behavior for Software Dynamic Analysis*", IEEE/ACM 8th International Workshop on Modeling in Software Engineering (MiSE),2016,Pages: 20 – 26,IEEE Conferences

[48] M P S Bhatia; Akshi Kumar; RohitBeniwal, "*Ontology based framework for reverse engineering of conventional softwares*",3rd International Conference on Computing for Sustainable Global Development (INDIACom) ,2016,Pages: 3645 – 3648,IEEE Conferences

[49] Andreas Sailer; Michael Deubzer; Gerald Lüttgen; Jürgen Mottok, "*CoreTAna: A Trace Analyzer for Reverse Engineering Real-Time Software*", IEEE 23rd International Conference on Software Analysis, *Evolution, and Reengineering (SANER)*,2016, Volume: 1Pages: 657 – 660,IEEE Conferences

[50] MaikelLeemans; Wil M. P. van der Aalst; Mark G. J. van den Brand, "*The Statechart Workbench: Enabling scalable software event log analysis using process mining*",2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER),Pages: 502 – 506,IEEE Conferences

[51] MaikelLeemans; Wil M. P. van der Aalst; Mark G. J. van den Brand, "*Recursion aware modeling and discovery for hierarchical software event log analysis*",2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER),Pages: 185 – 196,IEEE Conferences

[52] William Steingartner, "*Software support for course in semantics of programming languages*",2017 IEEE 14th International Scientific Conference on Informatics,Pages: 359 – 364,IEEE Conferences

[53] MasoudKargar; "*Semantic-based software clustering using hill climbing*"; 2017 International Symposium on Computer Science and Software Engineering Conference (CSSE), Pages: 55 – 60

[54] Asish Kumar Dalai; "*A code obfuscation technique to prevent reverse engineering*";2017 , (WiSPNET),Pages: 828 – 832

[55] Anshul Rani; "*Evolution of code smells over multiple versions of softwares: An empirical investigation*"; (I2CT) ;2017;Pages: 1093 – 1098,IEEE Conferences

[56] BernhardsBlumbergs; "*Bbuzz: A bit-aware fuzzing framework for network protocol systematic reverse engineering and analysis*"; (MILCOM); 2017;Pages: 707 – 712;IEEE Conferences

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

[58] EsaFauzi; BayuHendradjaya; "*Reverse engineering of source code to sequence diagram using abstract syntax tree*" ;2016 ,(ICoDSE);Pages: 1 – 6,IEEE Conferences

[59] RafalKudelski; Piotr Dudek; "*Using reverse engineering and rapid prototyping for patient specific orthoses*";(MEMSTECH), 2017',Pages: 88 – 90;IEEE Conferences

[60] Grant Williams; Anas Mahmoud; "*Mining Twitter Data for a More Responsive Software Engineering Process*";2017 -(ICSE-C);Pages: 280 – 282

[61] [V. Madhan; V. K. G. Kalaiselvi; Donald J. P; "*Tool development for formalizing the requirement for the safety criticalsoftware engineering process*";2017 2nd International Conference on Computing and Communications Technologies (ICCCT);Pages: 161 - 164

[62] Stephan Krusche; Bernd Bruegge; "*CSEPM - A Continuous Software Engineering Process Metamodel*"
2017 IEEE/ACM 3rd International Workshop on Rapid Continuous SoftwareEngineering (RCoSE);Pages: 2 – 8

[63] Muhammad Shafiq; Qinghua Zhang; "*Effect of project management in requirements engineering and requirements change management processes for global softwaredevelopment*"; IEEE Access;Pages: 1 – 1

[64] Rogelio Valdez-Almada; Oscar M. Rodriguez-Elias; "
Natural Language Processing and Text Mining to Identify Knowledge Profiles for Software Engineering Positions: Generating Knowledge Profiles from Resumes"
(CONISOFT);Pages: 97 - 106

[65] ISO/IEC/IEEE 15939: "*Iso/iec/ieee international standard - systems and software engineering--measurement process*";2017(E) - Redline;Pages: 1 - 100

[66] IEEE P16085/CD, "*IEEE Draft Standard - Systems and Software Engineering - Life CycleProcesses - Risk Management*" February 2018;Pages: 1 - 58

[67] Jameela Al-Jaroodi; Brandon Hollein; ; "*Applying software engineering processes for big data analytics applications development*";2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC);Pages: 1 – 7

[68] 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

[69] Hamza Hussein Altarturi; Keng-Yap Ng; "*A requirement engineering model for big data software*";2017 IEEE Conference on Big Data and Analytics (ICBDA);Pages: 111 – 117;IEEE Conferences

[70] Paul Ralph; "*Toward Methodological Guidelines for Process Theories and Taxonomies in Software Engineering*";IEEE Transactions on Software Engineering;Year: 2018, Pages: 1 - 1

[71] Software Quality assurance visited 15 April 2018, https://www.techopedia.com/definition/4363/software-quality-assurance-sqa

[72] IwanBinanto; Harco Leslie Hendric Spits Warnars . "*Measuring the quality of various version an object-oriented software utilizing CK metrics*". International Conference on Information and Communications Technology (2018): 41 - 44

[73] Syed Waqas Ali; QaziArbab Ahmed. "*Process to enhance the quality of software requirement specification document*".IEEE Conferences**.** International Conference on Engineering and Emerging Technologies (2018): 1 - 7

[74] 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

[75] 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

[76] 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

[77] 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

[78] Vinay Augustine; John Hudepohl."*Deploying Software Team Analytics in a Multinational Organization*" .IEEE Software (2018): 72 – 76,volume 35, Issue: 1

[79] Jehad Al Dallal; Anas Abdin. "Empirical Evaluation of the Impact of Object-Oriented Code Refactoring onQuality Attributes: A Systematic Literature Review".IEEE Transactions on Software Engineering(2018), Volume: 44, Issue: 1pages: 44 - 69

[80] 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

[81] Robert Chatley; Lawrence Jones. "Diggit: Automated code review via software repository mining".IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (2018): 567 – 571

[82] 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

[83] Eduard van der Bent; Jurriaan Hage. "How good is your puppet? An empirically defined and validated qualitymodel for puppet" .IEEE 25th International Conference on Software Analysis, Evolution and Reengineering(2018):164 – 174

[84] Md Rakibul Islam; Minhaz F. Zibran. "On the characteristics of buggy code clones: A code quality perspective". IEEE 12th International Workshop on Software Clones (2018): 23 – 29

[85] Ying Wang; Hai Yu; Zhiliang Zhu. "Automatic Software Refactoring via Weighted Clustering in Method-Level Networks". IEEE Transactions on Software Engineering (2018), Volume: 44, Issue: 3,Pages: 202 - 236

AUTHORS

First Author – Muhammad Muzammul, MS(Software Engineering), Softengr12@gmail.com, Government College University,Faisalabad,Pakistan