ISSN No. 0976-5697

Volume 4, No. 2, Jan-Feb 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Proposed Software Re-engineering Process That Combine Traditinal Software Re-engineering Process With Spiral Model

Ahmed Saleem Abbas*
Department of Computer Science & IT
Karbala University & SHIATS University Babylon, Iraq
ahmed_saleam@yahoo.com

DR. W. Jeberson
Department of Computer Science & IT
SHIATS University Allahabad, India
jeberson@rediffmail.com

V.V. Klinsega
Department of Computer Science & IT
SHIATS University Allahabad, India
segavijayakumar@rediffmail.com

Abstract: Software re-engineering, a recent research area includes reverse engineering, forward engineering and reengineering tools while re-engineering process modification to get new enhanced process until now does not discussed in a serious manner, so in this paper the traditional software reengineering process is discussed and the "proposed software reengineering process" is suggested. The proposed process incorporates both the reverse engineering process and forward engineering process integrated with spiral model where reverse engineering applies to existing system code to extract design & requirements, although this is often used as means to mitigate risks & reduced costs of operation and maintaining the software system. This paper briefly describes traditional re-engineering then discusses the emerging process of proposed-reengineering which is often used as means to simplify the complex tasks. The paper represents how quality is going to be effect with the help of given software reengineering process. An analysis of various possible risks, their impact and mapping with various attributes is correspondingly presented. This paper is also presenting the way to reduce the impact of most of these risks by using proposed re-engineering through get all the useful features of spiral model.

Keywords: Software Engineering, Software Re-engineering Process, Proposed Re-engineering Process, Reverses Engineering; Forward Engineering, Risk Assessment, Software Metrics.

I. INTRODUCTION

Re-engineering is the examination, analysis and alteration of an existing software system to reconstitute it in a new form, and the subsequent implementation of the new form [1]. Proposed Reengineering "is a re-engineering process that uses not just the traditional six activities, but a combination of these activities and a set of framework activities of the spiral model to transition an existing system to a target system.

There are a lot of number of risks associated with various principles such as Re-think & Re-specify, Re-Design, Re-Code, Re-Test i.e. Technical risk, known risk & project risk belonging to various sub categories of risk such as selection of code translation, operational, line by line translation is not possible, quality, reliability, external, interfacing of COTS with legacy system, schedule, COTS will perform up to the standards [2], because of these risks the thinking about planning and risk analysis activity of the spiral model is discussed as a solution to determine and reduce risks and their effects. The cyclic approach of spiral model is used within the proposed software reengineering to get the benefit of it;s incrementally growing a system degree of definition and implementation while it decreasing its degree of risk.

The process typically encompasses a combination of other processes such as reverse engineering, re-documentation, restructuring, translation and forward engineering. As the software industry moves to a new era, many new software design methodologies are developed, improving software reusability and maintainability, and decreasing development and maintenance time. But most companies have legacy systems that are costly to maintain. These systems cannot just be replaced with new systems. They contain corporate information and implied decisions that would be lost. They also are an investment, and were too costly to develop and evolve just to discard [2].

For these purposes, re-engineering becomes a useful tool to convert old, obsolete systems to more efficient, streamlined systems. But project development is always short on time and money, making the need to look at alternatives necessary. The use of COTS packages is seen as a way to increase reliability while decreasing development and test time. Translation of code is a means of decreasing time and cost. This has resulted in a combination of the development methods into a form of proposed re-engineering. In the proposed software re-engineering process there are a set of anchor point milestones for ensuring stakeholders commitment to feasible and mutually satisfactory system solution.

a. Re-engineering Model: The goal is to understand the existing software (specification, design and implementation) and then to re-implement it to improve the functionality, performance or implementation of the system and it is used to maintain the existing functionality and prepare for functionality to be added later, achieving greater

reliability, preparation for functional enhancement, improve maintainability & migration [1]

b. System Re-engineering: Re-structuring or re-writing part or all of a legacy system without changing its functionality considered as system re-engineering [3]. It applies where subsystem of a larger system require frequent maintenance [4]. Re-engineering involves adding effort to make functionality of those systems easier to maintain. The system may be restructured and re-documented [5]. Generally, there are following phases of software re-engineering, Source code translation, Reverse engineering, Program structure improvement, Program modularization and Data re-engineering.

II. RELATED WORK

Re-engineering is generally discussed as "business process change". Such change imposes new requirements on systems. Pooley et. al. include re-engineering in business process change not only changes over time within one organization but also the situation presenting many of the same problems in which a system developed in one organization and to be used in another[6]. Expert in re-engineering are much rarer than are experts in design and most of the engineers do not have much research experience in this area [5]. The problems with legacy systems had posed everywhere in the world. Brodie et. al. define a legacy system as one that significantly resists modification and evolution to meet new and constantly changing business requirements regardless of the technology used to design it [7]. The legacy system is replaced by a new system with the same or improved functionality [5].

In 2001, Alessandro Bianchi et al, they published paper titled "Iterative Reengineering of Legacy Functions", this paper describes a process of gradual reengineering of the procedural components of a legacy system. The proposed method enables the legacy system to be gradually emptied into the reengineered system, without needing to either duplicate the legacy system or freeze it. The process consists of evolving the legacy system components toward firstly a restored system and then toward the reengineered system. Meanwhile, the legacy system can coexist with both the restored and the reengineered parts. By the end of the process, a single system will be in existence: the reengineered one. The method has been applied to reengineer a real system and demonstrated its ability to: support gradual reengineering, maintain the system at work during the process, and minimize the need to freeze maintenance requests, renew the operative environment of the reengineered system with respect to the legacy system and, finally, eliminate all the system's aging symptoms This reengineering process is shown in figure 1, the various phases each being enclosed in a box, while the label on the arrow linking two successive phases indicates the output produced by the relative phase [8].

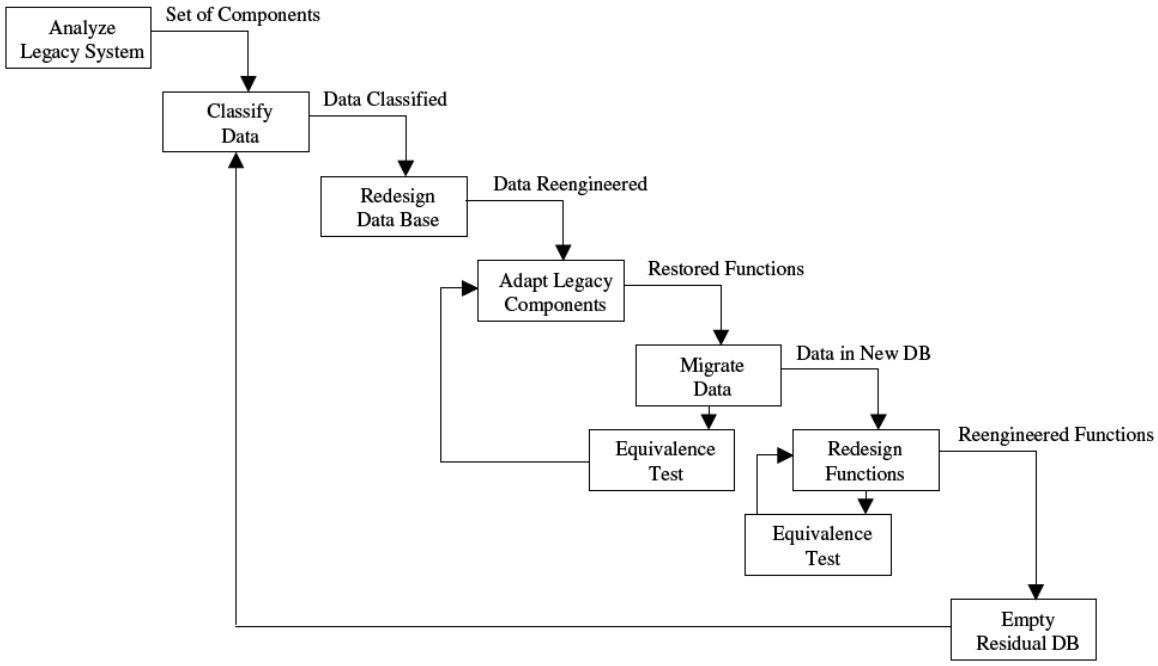

Figure 1. Iterative reengineering process [8].

In 2005, Xiaohu Yang et al. they published paper titled "A Dual-Spiral Reengineering Model for Legacy System "This paper presents a dual-spiral reengineering model, which performs as a cyclic approach. The main workflow in Dual-Spiral Reengineering Model requires that the two systems (legacy one and target one) work together, and move the functionality (not the modules) from the legacy system to the target system step by step, as in the spiral model. During the entire process, the active functionality in the legacy system is in a decremental pattern, and the active functionality in the new target system is in an incremental pattern, as Figure (2) [9].

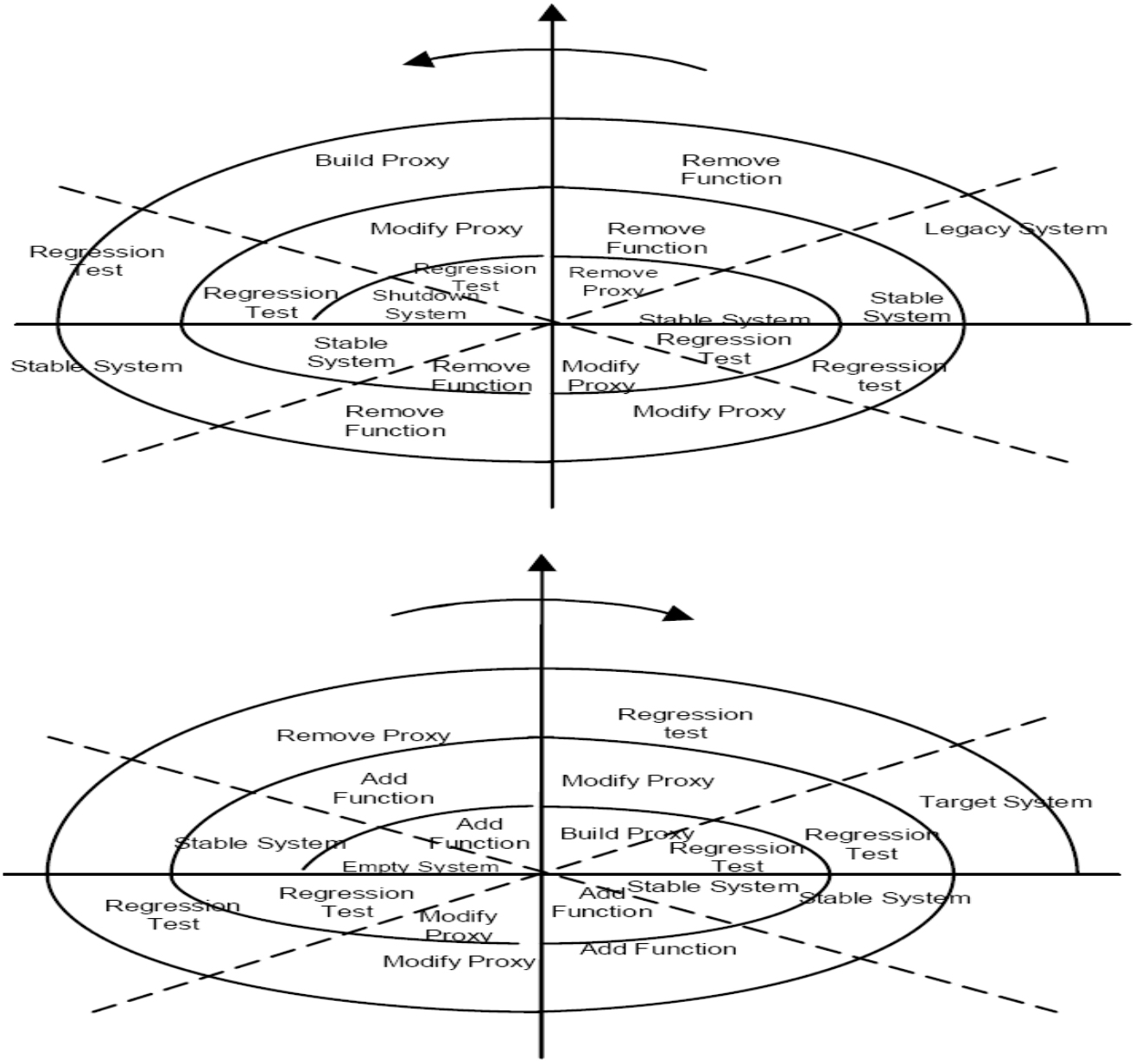

Figure 2. Decremental Legacy system Pattern and Incremental Target system Pattern in Dual-Spiral Model [9]

In 2012, Sandhya Tarar and Dr. Ela Kumar they design the Hybrid reengineering paradigm [2]. Hybrid re-engineering is a re-engineering process that uses not just a single, but a combination of abstraction levels and alteration methods to transition an existing system to a target system. In order to do the hybrid reengineering the reengineering model is mapped on with the software component library that can be Commercial off the shelf (COTS) or Math description engineering (MDE). Because hybrid reengineering uses COTS or MDE libraries through which design and requirements can be specified much faster thereby reducing effort, time and increasing reliability. COTS have some risk associated with it i.e. A package will not perform an anticipated or advertised, or that will be unreliable, immature or incomplete. Also COTS product may limit further enhancements to the system because changes in COTS provided functions may not be possible due to legal or contractual issues. All the main principles of hybrid reengineering approach namely re-specify, redesign, recode & retest the main task remains of integrating all the work of these principles together to make an effective model or system. Any specific principle is useless unless there is no integration with the other [2]. The Hybrid model is shown in figure (3).

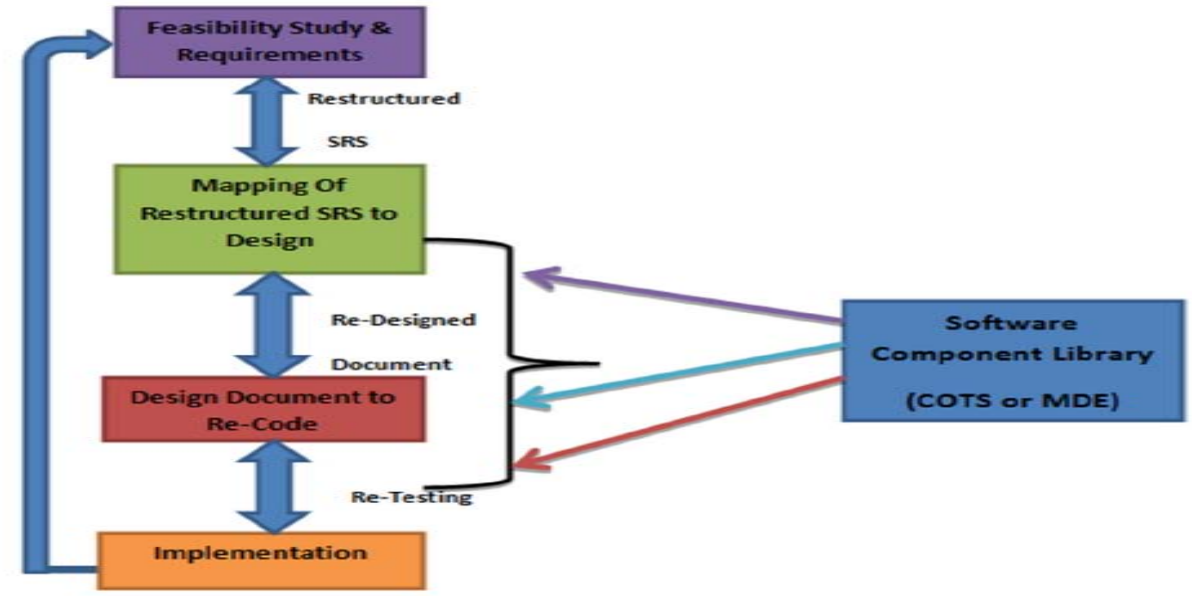

Figure 3. Hybrid Re-Engineering Model [2]

III. SPIRAL MODEL [10]

The spiral model is an evolutionary software process model that combines the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model. This model was shown in figure (4).

Spiral model is Prevalent in OO design methodologies, also encompasses all the essential development phases: Requirements analysis, Design, Code, Test and Maintenance. This model Marked by use of the control and structure of the more linear, sequential waterfall process, but with a series of "evolutionary releases", it is also known as the "hybrid model", Explicitly addresses the issue of quality assurance by performing the development process in a "step-wise refinement" method. Each step produces a "deliverable" which embodies the structure or sequential nature of the process. This process naturally focuses on a high degree of customer or stakeholder involvement during the development process.

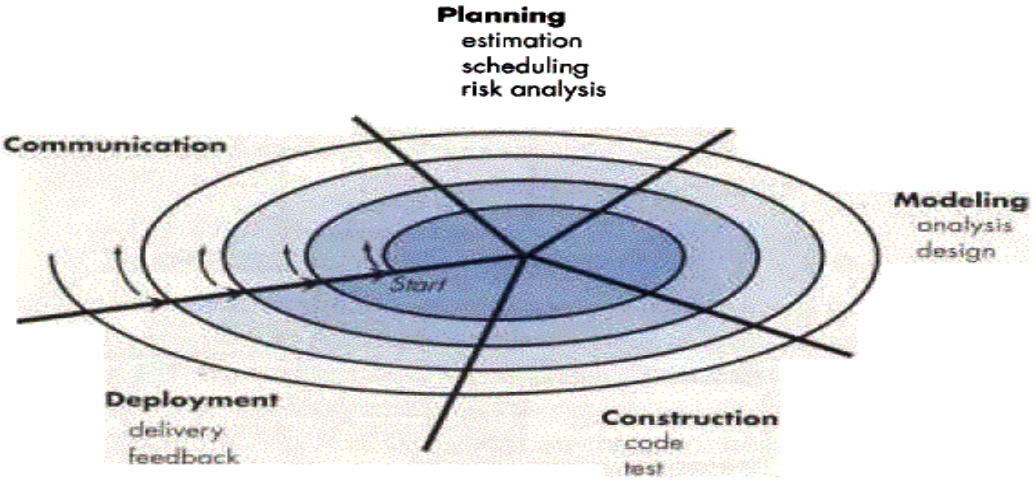

Figure 4. spiral model [10].

In this diagram, "Communication" refers to the Requirements and analysis process, "Planning" corresponds to preliminary design and scheduling, "Modeling" is the detailed design, "Construction" refers to code/debug/integration, and "Deployment" is the delivery to the customer and the feedback process. Software is developed in a series of evolutionary releases

a. During early iterations, the release might be a paper model or prototype
b. During later iterations, increasingly more complete versions of the engineered system are produced

c. The final iteration produces the complete software product

First circuit around the spiral might result in the development of the product specification; might result in a CoDR review by the customer. Next iteration might produce a prototype, containing the GUI, for example; the customer might want to see this, so there could be a PDR and/or CDR at this time. Third time around might be used to fill in more detailed functionality, and release a preliminary working model. Fourth circuit might result in a complete alpha release, which the customer could "hammer on" for a while to test robustness and provide feedback to the solution provider about the product's strengths and weaknesses. Fifth iteration might be a beta test, or it could be the final build for initial release (if the previous circuit was satisfactory enough to warrant this).

IV. TRADITIONAL SOFTWARE REENGINEERING PROCESS:[10]

Reengineering takes time; it costs significant amounts of money; and it absorbs resources that might be otherwise occupied on immediate concerns. For all of these reasons, reengineering is not accomplished in a few months or even a few years. Reengineering of information systems is an activity that will absorb information technology resources for many years. That's why every organization needs a pragmatic strategy for software reengineering. A workable strategy is encompassed in a reengineering process model.. Reengineering is a rebuilding activity, to implement it apply a software reengineering process model that defines six activities, shown in Figure (5). In some cases, these activities occur in a linear sequence, but this is not always the case. For example, it may be that reverse engineering (understanding the internal workings of a program) may have to occur before document restructuring can commence. The reengineering paradigm shown in the figure is a cyclical model. This means that each of the activities presented as a part of the paradigm may be revisited. For any particular cycle, the process can terminate after any one of these activities.

a. ***Inventory analysis:*** Every software organization should have an inventory of all applications. The inventory can be nothing more than a spreadsheet model containing information that provides a detailed description (e.g., size, age, business criticality) of every active application.

b. ***Document Restructuring****:* Weak documentation is the trademark of many legacy systems. But What are available options to solve it?

(a). Creating documentation is far too time consuming. If the system works, we'll live with what we have.

(b). Documentation must be updated, but there are limited resources. We'll use a "document when touched" approach.

(c). The system is business critical and must be fully redocumented.

c. ***Reverse Engineering****:* the Reverse engineering is the process of analyzing a program in an effort to create a representation of the program at a higher level of abstraction than source code. Reverse engineering is a process of design recovery. Reverse engineering tools extract data, architectural, and procedural design information from an existing program.

d. ***Code Restructuring****:* The most common type of reengineering (actually, the use of the term reengineering is questionable in this case) is code restructuring. Some legacy systems have relatively solid program architecture, but individual modules were coded in a way that makes them difficult to understand, test, and maintain. In such cases, the code within the suspect modules can be restructured. To accomplish this activity, the source code is analyzed using a restructuring tool. Violations of structured programming constructs are noted and code is then restructured (this can be done automatically). The resultant restructured code is reviewed and tested to ensure that no anomalies have been introduced. Internal code documentation is updated.

e. ***Data Restructuring:*** A program with weak data architecture will be difficult to adapt and enhance. In fact, for many applications, data architecture has more to do with the long-term viability of a program that the source code itself. Unlike code restructuring, which occurs at a relatively low level of abstraction, data structuring is a full-scale reengineering activity. In most cases, data restructuring begins with a reverse engineering activity. Data objects and attributes are identified, and existing data structures are reviewed for quality. When data structure is weak (e.g., flat files are currently implemented, when a relational approach would greatly simplify processing), the data are reengineered. Because data architecture has a strong influence on program architecture and the algorithms that populate it, changes to the data will invariably result in either architectural or code-level changes.

f. ***Forward Engineering****:* In an ideal world, applications would be rebuilt using a automated "reengineering engine." The old program would be fed into the engine, analyzed, restructured, and then regenerated in a form that exhibited the best aspects of software quality. In the short term, it is unlikely that such an "engine" will appear, but CASE vendors have introduced tools that provide a limited subset of these capabilities that addresses specific application domains (e.g., applications that are implemented using a specific database system). More important, these reengineering tools are becoming increasingly more sophisticated.

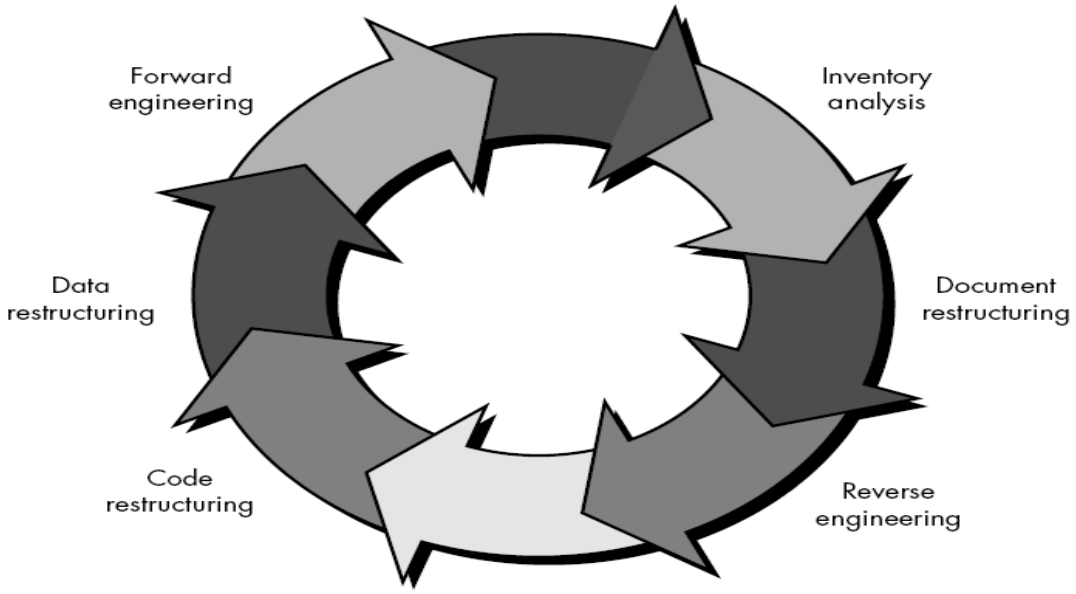

Figure 5. A software reengineering process model [10]

V. PROPOSED SOFTWARE REENGINEERING PROCESS

In this section a new reengineering process was suggested and named as a "Proposed Reengineering Process". It is a re-engineering process that uses not just the six activities of traditional re-engineering process, but a combination of these activities and a set of framework activities of the spiral model to perform the transition of an existing system (legacy system) to a target system (new system).The proposed reengineering process consists of eight task regions, these tasks regions as follows:

a. ***Communication & Detection the Frailer Reasons of Legacy SW:-*** Communication with customer in order to re-specify the requirements as per the new need of the user, and then map these requirements with the SRS. The frailer reasons of the selected legacy software were detected within this first step and the feasibility study is done.

b. ***Planning:-*** it is the activity that required to define resources, timelines, cost and other project related information, then risk analysis is accomplished to assess both technical and management risks.

c. ***Reverse Engineering:-*** Reverse engineering is the process that starts from the implementation phase and moving towards the coding, design and requirement phase, through this region the data, architectural and procedural design information was extracted from existing legacy software.

d. ***Document Restructures:*** in this phase the mapping of restructured SRS to the Design document is being done i.e. integration of new SRS to design in order to get the redesigned document which is the output of this phase. As SRS changes the design structure consisting of DFD/ ER diagrams/UML diagrams need to be changed as per the extent of changed requirements.

e. ***Code Restructures:*** To accomplish this activity, the source code is analyzed using a restructuring tool. The resultant restructured code is reviewed and tested to ensure that no Distortions have been introduced. Internal code documentation is updated.

f. ***Data Restructures:*** In most cases, data restructuring begins with a reverse engineering activity. Data objects and attributes are identified, and existing data structures are reviewed for quality. When data structure is weak, the data are reengineered. Because data architecture has a strong influence on program architecture and the algorithms that populate it, changes to the data will invariably result in either architectural or code-level changes.

g. ***Forward Engineering:*** it is the process which starts from requirement to the implementation phase. Forward engineering, also called renovation or reclamation, not only recovers design information from existing software, but uses this information to alter or reconstitute the existing system in an effort to improve its overall quality. In most cases, reengineered software re-implements the function of the existing system and also adds new functions and/or improves overall performance.

h. ***Deployment:*** it is the delivery of reengineered software to the customer and then feedback process is done to obtain customer evaluation of that software.

These eight task regions are shown in the (figure 6) and they are very important to reengineer the legacy software; this proposed process model integrated the iteration nature of spiral model with traditional software reengineering process. There are basics task that have to be accomplished

by any software reengineering process but the diagnostic features of the proposed process are the following:

a) The millstones are inserted after each iteration to check if the process is performed within schedule time or not.

b) The reengineering process can be finished after any iteration.

c) The reasons that cause the failure of the legacy software are determined at first step to remedy these reasons.

d) In the proposed reengineering process there is planning task regions that make feasibility study, scheduling, estimation and risk analysis to determine all the expected benefit and estimation of the required cost and effort, so according this the developer can make his decision to reengineer that legacy software or rebuild it from scratch.

e) The third, fourth, fifth, sixth, seventh are the same of that activities that done in traditional software reengineering process except that in the proposed there are a non determined number of iterations that allowed to reach to the final project with new features, capabilities and high quality software.

f) Deployment and waiting for the customer feedback it is also diagnostic activity of the proposed reengineering process. According this feedback the decision is made if the reengineering process is finished or not. If the customer dose not satisfy the next iteration will began.Authors and Affiliations

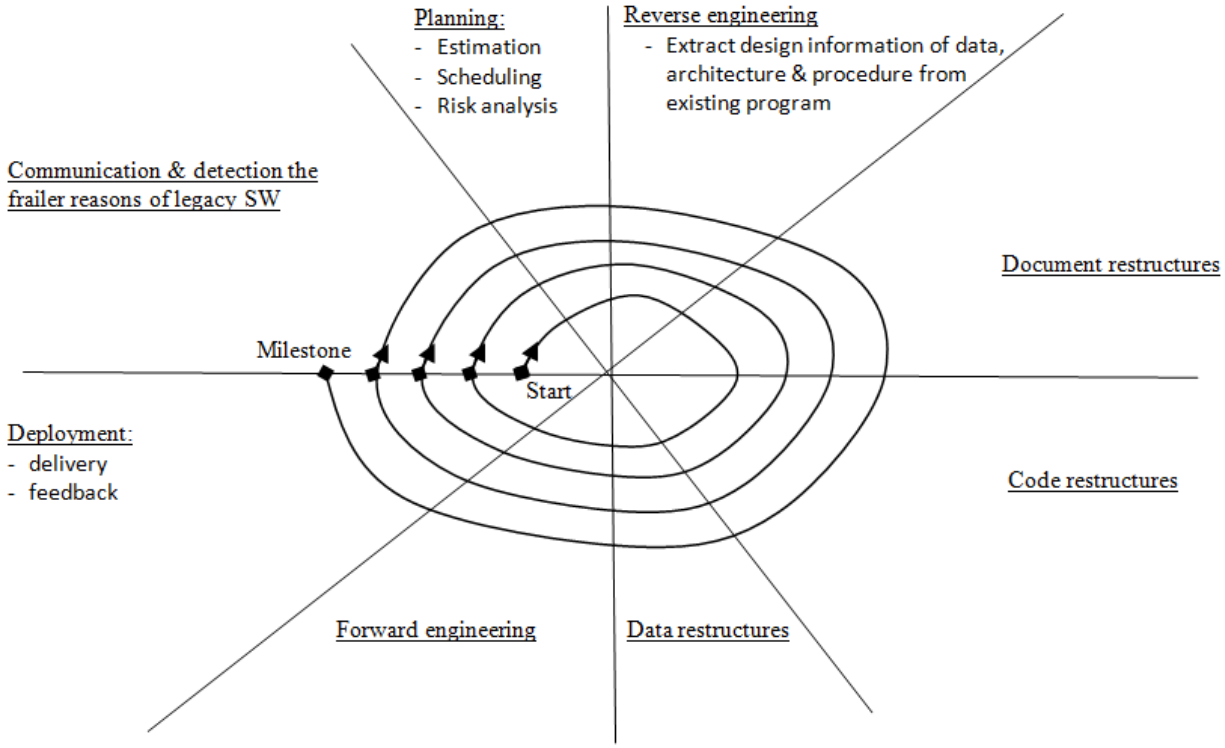

Figure 6: Proposed reengineering process

A. Benefits of Proposed software reengineering process:

Proposed re-engineering has the advantage that it determine the causes of failure to overcome it and prevent it for the future, also it is reduces the development schedule time and reduced the costs. This process can be used to produce many versions of modern software by using the legacy software as a baseline.

B. Limitations of Proposed software reengineering process:

Since the proposed re-engineering is a new approach in the re-engineering area although a very cost efficient approach as it reduces development time & cost but there are no exact metrics available for this in order to measure the scalability & performance and most of these metrics depend on estimation values such as human effort (person/ hour), this metric is deferent from person to other according to experience and skills of that person, also cost and time are depend on human effort metrics and the complexity of the software, so this is a main limitation that face as a developer.

VI. EXPECTED RISKS

There are number of risk that can be recognized and the reengineering teams have to take care about them, they are:

(a). Development Environment: risks associated the availability and quality of the COTS/MDE tools.
(b). Product Size: risks impact is directly proportional to the overall product size that has to be re-engineered/re-build/re-modulated.
(c). Size and experience; risks associated with the overall technical and project experience of the software engineers who will do the task of re-engineering.
(d). Customer characteristics: risks associated with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.
(e). Technology to be built: risks associated with the complexity of the system to be reengineered and the "newness" of the technology that is packaged by the system.
(f). Process definition: risks associated with respect to the process followed by the organization to perform re-engineering process.
(g). Business impact: risks associated with constraints imposed by management or the marketplace and also the Domain of the business causes risks for performing reengineering process.

The reengineering risk can be distributed into six risk areas as shown in figure (7). These risk areas as following [11]:

a. Technology risks:

a) Recovered information is not useful or used

b) Reverse engineering to representations that cannot be shared

c) Reengineering technology inadequate to accomplish reengineering goals

b. Tool risks:

a) Dependence on tools that do not perform as advertised

b) Not using installed tools

c. Strategy risks:

a) Premature commitment to a reengineering solution for an entire system

b) Failure to have a long-term vision with interim goals

c) Lack of global view: code, data, process reengineering

d) No plan for using reengineering tools

d. Application risks:

a) Reengineering with no local application experts available

b) Existing business knowledge embedded in source code is lost

c) Reengineered system does not perform adequately

e. Process risks

a) Extremely high manual reengineering costs

b) Cost benefits not realized in required time frame

c) Cannot economically justify the reengineering effort

d) Reengineering effort drifts

e) Lack of management commitment to ongoing reengineering solution

f. Personnel risks:

a) Programmers inhibiting the start of reengineering

b) Programmers performing less effectively to make an unpopular reengineering project look less effective

Figure (7): Re-engineering Risks area [11]

VII. SOFTWARE METRICS

Metrics is a quantitative measurement of its attributes. Various parameters such as Cost, Reliability, Quality, Object-Oriented Measures, Reusability, Complexity, and Portability can be used to measure the software re-engineering in quantitative form. As shown in table (1) corresponding metrics leads to various factors responsible for the measurement of the software re-engineering [2][10].

Table I. Software Reengineering Metrics

Complexity	• Size (LOC,FP) • Design structure • Data analysis
Cost	• Effort (person, months) • Time • Productivity
Reliability	• Interoperability • Availability • MTTR • MTTF
Reusability	• Retrievability • Integrability • Testability • Generality • Portability • Modifiability
Quality	• Functionality • DRE • Complexity • Reliability • Reusability • Performance • Security
Object oriented measures	• Inheritance • Encapsulation • Polymorphism • Abstraction

VIII. CONCLUSION

The new era is recognized by the very fast changes in the computer industry that introduce new hardware and software, making older systems suffers from the market competition and the difficulty in maintenance. Software re-engineering provides reduced risk level. There is a high risk in new software

development. There may be development problems, staffing problems and specification problems which are reduced by the use of reengineering process. Another advantage of software re-engineering is reduced cost. The cost of re-engineering is often significantly less than the costs of developing new software because some part i.e. sub-systems are used as it is and some sub-systems are changed. Based on previous history, project development is usually short on time and money, making it necessary to look at another solution. "Proposed Re-engineering" plays as a solution in this scenario. The use of re-engineering approaches provides a way to increase reliability and quality of the system while decreasing development efforts. Translation of code is a means of decreasing time and cost. This results in a combination of development methods of traditional software reengineering with the Spiral software development model into a form of proposed re-engineering process model.

IX. ACKNOWLEDGMENT

We thank everyone at the SHIATS/department of computer science and information technology who participated in this research for many stimulating discussions. Also special thanks to Karbala University/department of computer science for them great support.

X. REFERENCES

[1] L Manzella. Mutafelija.: Concept of re-engineering Life Cycle, ICSI Second International Conference On System Integration. IEEE. 1992.

[2] Sandhya Tarar, Dr. Ela Kumar, "Design Paradigm and Risk Assessment of Hybrid Re-engineering with an approach for development of Re-engineering Metrics", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.

[3] Linda H. Rosen Berg, Lawrence E.Haytt. : Hybrid Re-engineering, Software technology conference. Utah, April 1997.

[4] MindTree Ltd.: Rehosting & Re-engineering from Mainframe to Wintel, Case Study, 2010.

[5] Shekhar Singh, Significant role of COTS to design Software Reengineering Patterns, International Conference on Software Engineering and Applications(ICSEA),2009.

[6] 15. Pooley R., Stevens P., Systems Reengineering Patterns, CSG internal report, 1998.

[7] Brodie, Michael L., and Stonebraker, Michael, “Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach”, Morgan-Kaufman Publishers, 1995.

[8] Alessandro Bianchi et al, " Iterative Reengineering of Legacy Functions", IEEE International Conference on ISBN: 07695118 Year: 2001 Pages: 632-641 Provider: IEEE Publisher: IEEE.

[9] Xiaohu Yang et al, "A Dual-Spiral Reengineering Model for Legacy System", TENCON 2005 - 2005 IEEE Region 10 Conference ISBN: 0780393112 Year: 2005 Pages: 1-5 Provider: IEEE Publisher: IEEE.

[10] Pressman, Scott (2005), Software Engineering: A Practitioner's Approach (Sixth, International ed.), McGraw-Hill Education Pressman.

[11] Ian Sommerville, “Software Engineering, 6th edition”, 2000.

