International Journal of Computer Science
and Engineering (IJCSE)
ISSN 2278-9960
Vol. 2, Issue 2, May 2013, 53-64

International Academy of Science,
Engineering and Technology
Connecting Researchers; Nurturing Innovations

IMPLEMANTATION OF FUSION MODEL TO RE-ENGINEER LEGACY SOFTWARE

AHMED SALEEM ABBAS[1], W. JEBERSON[2] & V. V. KLINSEGA[3]

[1]Doing Ph.D at SHIATS & Assistant Lecturer at Department of Computer Science, Karbala University, Iraq

[2,3]Associate Professor, Department of Computer Science & IT, SHIATS University, Allahabad, India

ABSTRACT

Software re-engineering, is a recent research area which includes reverse engineering, forward engineering and reengineering. In this paper, the software reengineering process model proposed by the same authors in another paper titled "Proposed Software Re-engineering Process That Combine Traditional Software Re-engineering Process with Spiral Model" [1] is validated by implementing it on a legacy system, and its name is renamed as FUSION model. This new model 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 & reduce operation and maintenance cost of the software system. This paper briefly describes the FUSION model which can be used to simplify the complex tasks. The paper represents how the legacy supermarket management system can be re-engineered using the new approach then the result of comparison between the legacy software and the re-engineered software will be presented. The features of the FUSION model are also presented in later section of the paper.

KEYWORDS: Cyclomatic Complexity, Quality Metrics, Reverse Engineering, Risk Assessment, Software Engineering, Software Re-Engineering Process

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 [2]. FUSION Reengineering model "is a re-engineering process that uses not just the traditional six activities. (Inventory analysis, Document restructuring, Reverse engineering, Code restructuring, Data restructuring, Forward engineering), but a combination of these activities and a set of framework activities of the spiral model to transform an existing system to a target system.

There are a 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 [3], because of these risks the planning and risk analysis activity of the spiral model is integrated with FUSION model as a solution to determine and minimize risks. The cyclic approach of spiral model is used within the FUSION software reengineering model to get the benefit of its incrementally growing system degree of definition and implementation while decreasing the degree of risk.

As the software industry moves to a new era, many new software design methodologies are being 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 [3].

For these purposes, re-engineering becomes a useful tool to convert old, obsolete systems to more efficient,

streamlined systems. FUSION model is used to convert the legacy supermarket management system that was developed by using visual basic 6 to the target system according to the feedback of the customer, where the legacy system was used as a base line for this reengineering process. In the next sections a quick review of the basic concepts of reengineering will be done. After that the FUSION model will be discussed step by step and how these steps were applied on the supermarket legacy system, which will be followed by a discussion on the tools that helped us to apply and evaluate this proposed reengineering process.

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[5]. 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 [4]. 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 [6]. The legacy system is replaced by a new system with the same or improved functionality [4].

The paper published by Alessandro Bianchi et al on Iterative Reengineering of Legacy Functions 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 having the need to either duplicate the legacy system or freeze it. The process consists of evolving the legacy system components firstly towards 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, which is 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 [7].

A Dual-Spiral Reengineering Model for Legacy System proposed by Xiaohu Yang et al. 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 [8].

The Hybrid reengineering paradigm [3] proposed by Sandhya Tarar and Dr. Ela Kumar. Hybrid re-engineering is a re-engineering process that uses not just a single, but a combination of abstraction levels and alteration methods to transform an existing system to a target system. Here 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. The main task in hybrid reengineering is to integrate its main approach namely re-specifying, redesigning, recoding & retesting to work together to make an effective model or system. Any specific principle is useless unless there is no integration with the other [3].

BASIC CONCEPTS OF FUSION MODEL

The FUSION model enables cyclic reengineering of a legacy system, where the various components will go through different states during this process. Each component may be in one of the following states during the various steps of the reengineering process:

- Legacy, i.e. components of the legacy system that have not yet been reengineered.
- Restored, i.e. functions with the same structure as in the legacy system, but access data through the new data banker, that alone recognizes the physical structure of the database.
- Reengineered, i.e. components of the legacy system that have already been modified, and whose quality has reached the desired levels.
- New, i.e. components that did not exist in the legacy system, but have been added to introduce new functions in the same application domain.

The FUSION model is a re-engineering process model 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). As shown in the figure 1, there are eight task regions, which were applied by the software engineering team on legacy software to get the target re-engineered software, for this purpose we have chosen a legacy software known as supermarket management system that was developed in visual basic before many years and all the steps required to convert it to the target system will be discussed as follows:

- **Identification Failure Reasons of Legacy SW:** The software re-engineering teem should meet with customer in order to re-specify the requirements as per the current expectations of the user. In the supermarket system the new requirements are identified such as convert the system from windows based to web based, create two levels of authentication as adminstrator level and employee level, add the facility to generate many types of reports about the poduct, stock master, supplier, sale and purchase information, and feedback report, followed by mapping these requirements with the SRS. The reasons for failure of the selected legacy software were detected within this first step, some of which are that the legacy software don't have any type of security, there is weak error handling procedure that does not cover all the possible error cases, and that the old system can work on only one computer which results in poor service with only one counter in the supermarket to generate the bill for the customer. After that, the feasibility study was conducted,

In this sense we express the *mean time to maintenance request* for the i-th component ($MTTMR_i$) with the following formula:

$$MTTMR_i = \sum_{j=1}^{n} \frac{\Delta t_{j,i}}{n_i}$$

, where:

- ni is the total number of requests for maintenance made for the i-th component;
- $\Delta t_{j,i} = t_{j,i} - t_{j-1,i}$ is the time interval between two successive requests for maintenance of the i-th component.

The values for n_i and $\Delta t_{j,\,I}$ are obtained from the historical archives of the system. When the expected time for reengineering the i-th component, RT_i, is less than $MTTMR_i$, then it is reasonable to suppose that it is unlikely to get requests for maintenance of this component during the time it is being reengineered. On vice versa, if RT_i, is higher than $MTTMR_i$, then the component must be subdivided into several subcomponents, to prevent the requests for maintenance that are likely to be made during this time from being held up for very long time, and according to this analysis the decision to continue in re-engineering process on the legacy software was made

- **Planning:** it is the activity of defining resources, timelines, cost, and other project related information, to carry out the re-engineering process. It was decided to use ASP.net with C# programming language to design and implement all web forms of the system, and to use Microsoft SQL server 2008 to redesign the database and implement it. The duration of the whole process was estimated to be three months, and the cost benefit of re-engineering was calculated according to Sneed [9] formula, followed by risk analysis which was accomplished to assess both technical and management risks.

- **Reverse Engineering:** Reverse engineering is the process that starts from the implementation phase and moving towards the coding, design and requirement phase, as a result of which the data, architectural and procedural design information was extracted from existing legacy software. In this case study the "Visustin v7" analysis tool was used to Visualize the code of the legacy supermarket manangemant system with flow charts! Visustin converts source code to flow charts and UML activity diagrams to understand existing code.

- **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, the algorithms have to be reviewed and modified and program logic should be verified. In this case study the "Visustin v7" analysis tool was used to restructure the document of complex functions of legacy software.

- **Code Restructures:** To accomplish this activity, we used "Visustin v7" restructuring tool and the resultant restructured code is reviewed and tested to ensure that no distortions have been introduced and the internal code documentation is updated.

- **Data Restructures:** In most cases, data restructuring begins with a reverse engineering activity. Data objects and attributes are identified, existing data structures are reviewed for quality and the data are reengineered when data structure is weak, 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. In this case study all varaibles are reviewed and we found that the data structure was weak and so we restructured it, adding some new variables to be used in the new functions. From a conceptual point of view, the legacy data are partitioned into two classes:

- Essential data, needed to carry out the application's business functions; The essential data contain:

  - Conceptual data, that is specific to the application domain, and for which they have a precise conceptual meaning;

  - Necessary structural data, that do not belong to the above class but constitute the primary keys to the tables making up the database.

- Residual data, that are not necessary for carrying out the business functions, but are used by the legacy system and must therefore remain in the database until the procedures that use them have been reengineered. Instead, the residual data class contains all the data that existed in the legacy database, and which should ultimately be eliminated to improve the software quality. They are classified as:
    - Control data, that communicate to a procedure that an event occurred during execution of other procedures, thus controlling the behavior of the former one.
    - Redundant structural data, used to organize and support the data structures of the legacy system, but which are not strictly required, and can be removed with a better design of the database.
    - Semantically redundant data, whose definition domain is the same as, or is contained in, the definition domain of other data, while each equal value in the two definition domains is interpreted in the same way.
    - Computationally redundant *data* which can be computed starting from a different set of data included in the same database.

In this case study there are seven tables in the database of the legacy supermarket management system they are customer bill, customer detail, department, employee, items, transaction, and vendor and there are three tables were added to the reengineered database as per the customer need which are user login, product master, and feedback.

- 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 functions of the existing system and also adds new functions to improve overall performance. In this case study we use a Rapid Application Development tool, Microsoft visual studio 2008 with C# and some web site templates to renovate the selected legacy software.
- Deployment: It is the delivery of reengineered software to the customer. Before this step the system should be tested by using any testing technique. But with the proposed software reengineering process the recommended testing techniques are Alpha testing and then Beta testing which is conducted when the project is delivered to the customer. The feedback process is necessory to obtain customer evaluation of that software, according to which the re-engineering process iterates to meet the customer demand.

The eight task regions are shown in the (figure 1) and they are very important to reengineer the legacy software. The FUSION model integrates the iteration nature of spiral model with traditional software reengineering process.

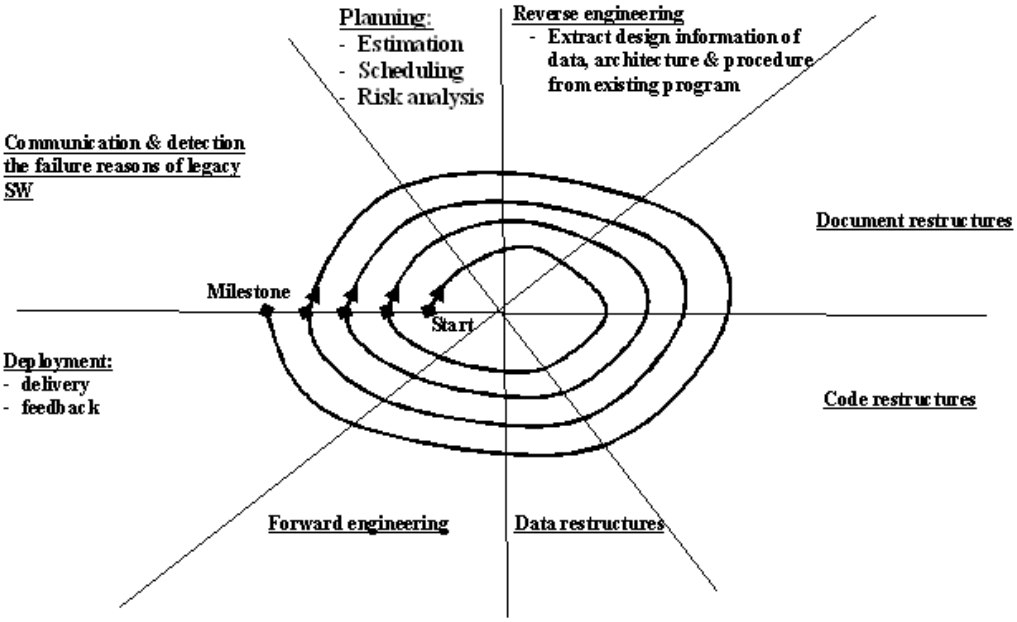

Figure 1: FUSION Reengineering Process

DIFFERENCES BETWEEN FUSION RE-ENGINEERING MODEL AND OTHER MODELS

There are many basic tasks that have to be accomplished by any software reengineering process but the diagnostic features of the proposed process are the following:

- The milestones are inserted after each iteration to check if the process is performed within scheduled time or not.
- The reengineering process can stop after any iteration.
- The reasons that cause the failure of the legacy software are determined at first step to remedy these reasons.
- In the proposed reengineering process there is planning task that includes feasibility study, scheduling, estimation and risk analysis to determine the expected benefit and estimation of the cost and effort, so that the developer can make his decision on whether to reengineer that legacy software or rebuild it from scratch.
- The third, fourth, fifth, sixth and seventh activities are the same as in traditional software reengineering process except that in the proposed model there is a non deterministic number of iterations that lead to the final reengineered product with new features, capabilities and high quality.
- Deployment and waiting for customer feedback is also a diagnostic activity of the proposed reengineering process. According to this feedback the decision is made if the reengineering process is completed or not. If the customer is not satisfied the next iteration will begin.

BENEFITS OF PROPOSED SOFTWARE REENGINEERING PROCESS AND THE COST BENEFIT FORMULA

Proposed re-engineering model has the advantage that it determines the causes of failure before any decision can be made, in order to overcome and prevent the failures in the future which in turn reduces the development time and cost. This model can be used to produce many versions of modern software by using the legacy software as a baseline.

According to the cost/benefit analysis model for reengineering, proposed by Sneed [9], the cost benefit for the proposed model can be calculated with respect to nine parameters as defined below:

P_1= current annual maintenance cost for an application

P_2= current annual operation cost for an application

P_3= current annual business value of an application

P_4= predicted annual maintenance cost after reengineering

P_5= predicted annual operations cost after reengineering

P_6= predicted annual business value after reengineering

P_7= estimated reengineering costs

P_8= estimated reengineering calendar time

P_9= reengineering risk factor (P_9 = 1.0 is nominal)

L = expected life of the system

From the above, the following cost values can be obtained:

- (C_{maint}) the cost associated with the continuous maintenance of a candidate application (i.e. if reengineering is not performed) defined as

  $C_{maint} = [P_3 – (P_1 + P_2)] * L$

- (Creeng) that is the cost associated with reengineering a candidate application defined as

  $C_{reeng} = [P_6 – (P_4 + P_5)] * (L – P_8) - (P_7 * P_9)$

- Using equations 1 and 2 above, the overall benefit of reengineering can be computed thus:

  Cost-benefit = $C_{reeng} - C_{maint}$

For a software organization, the cost-benefit presented above can be performed for all high priority applications identified during inventory analysis and those that show high cost-benefit can be targeted for reengineering, while others can be postponed until resources are available. The model does not specify how to compute P_4 to P_9 hence relies on costs estimation models like Constructive Cost Model II (COCOMO II), Halstead, Expert Judgment, Function Point, and Lines-of-Code.

Those applications that show the highest cost/benefit can be targeted for reengineering, while work on others can be postponed until resources are available.

In this case study the predicted parameters to calculate the cost benefit of reengineering the legacy supermarket system are as follows:

P_1= 400$, P_2= 150$, P_3= 600$,

P_4= 100$, P_5= 100$, P_6= 1500$,

P_7= 150$, P_8= 3 months= 0.25 years,

P_9= 1, L = 4 years.

The cost associated with the continuous maintenance of the legacy supermarket system before reengineering is equal to 200$ and the cost associated with reengineering of the legacy supermarket system is equal to 600$, and the overall benefit of reengineering can be computed from these two values and it is equal to 400$.

EXPECTED RISKS

There are a number of risks that can be recognized and the reengineering teams have to take care of them, some of which are:

- **Development Environment:** Risks may be associated with the availability and quality of COTS (Common-Off-The-Shelf) tools. In this case study such risks are minimized by using Asp.net tools as RAD (Rapid application development) from Microsoft, given that the Microsoft tools are most popular and trusted tools.
- **Product Size:** Impact of risks is directly proportional to the overall size of the product that has to be re-engineered/re-build/re-modulated. In this case study the size of the legacy software is 2411 SLOC (Source Line of Code) and the number of modules is 212.
- **Experience:** Risks associated with the overall technical and project experience of the software engineers who are involved in the process of re-engineering. In this case study this risk was eliminated by the engineers taking advance technical course in ASP.net and SQL server.
- **Customer Characteristics:** Risks associated with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.
- **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. The complexity of the system can be determined through the Cyclomatic Complexity, which it is a measurement of complexity of a program code and can be calculated by using the control flow graph of the program. In this case study the total Cyclomatic complexity of the legacy supermarket management system is 578, the Mean Cyclomatic complexity is 2.726, and the decision density is 0.2397. All these values are calculated by using “Visustin v7” analysis tool and according to the following formulas:
  - Code Cyclomatic Complexity = Number of graph edges – Number of Graph Nodes + Number of connected components
  - Mean Cyclomatic complexity= Cyclomatic Complexity / Number of modules
  - Decision density= Cyclomatic complexity / Number of lines

The calculated values show that, the legacy supermarket software is a simple program, without much risk. The following table (table 1), displays the results published by SEI (Software Engineering Institute) and they are being used widely to determine the health of the code base and to evaluate the risk associated with the application [10].

Table 1: Risk Evaluation According to Mean Cyclomatic Complexity

Mean Cyclomatic Complexity	Risk Evaluation
1-10	A simple program, without much risk
11-20	More complex, moderate risk
21-50	Complex, high risk program
Greater than 50	Un testable program (very high risk)

- **Process Definition:** Risks associated with respect to the process followed by the organization to perform re-engineering process. In this case study the used model for reengineering is the FUSION model that has the cyclic features and that depends on the feedback of the customer obtained at the final step of each iteration to evaluate and regenerate next versions of the reengineered system, reducing the risks.
- **Business Impact:** The constraints imposed by management or the market place and also the domain of the business are likely to cause risks in performing reengineering process. In this case such risks are minimized through fine planning at the planning phase.

RESULTS AND DISCUSSIONS

In this paper the FUSION reengineering model is applied on legacy supermarket management system to convert it to a new system with more functionality, high security, and good quality. The FUSION reengineering model is applied to get and prove the benefit of reengineering, and the software metrics is used to make comparison between the legacy system and new system, and the result of comparison is shown in table (2). Some metrics are calculated by using "Visustin v7" analysis tool such as Size (LOC), Number of modules, Cyclomatic Complexity, Mean Cyclomatic Complexity, and Decision Density.

Three levels are used to evaluate the metrics and these levels are Low (L), Medium (M), and High (H). Result of technical comparison between legacy supermarket management system and the re-engineered system is shown in Table (3) and this comparison includes the difference in Platform, Application, Programming Language, Front end, Backend, Database, Number of tables, Number of generated reports, Execution time, Coding, Making dll (dynamic link library), Authentication levels, Stock management and Customer Feedback.

From the data in table (2) and table (3), it is proved that the reengineered system is better than the legacy system and it meets the customer need.

Table 2: Metrics Comparision between Legacy Supermarket Management System and the Re-Engineered System

		Legacy Supermarket Management System	Re-Engineered Supermarket Management System
Complexity	Size (LOC)	2411	2323
	Number of modules	212	139
	Cyclomatic Complexity	578	354
	Mean Cyclomatic complexity	2.726	2.546
	Decision density	0.239	0.152
Reusability	Integrability	L **M** H	L **M** H
	Testability	L **M** H	L M **H**
	Portability	**L** M H	L M **H**
	Modifiability	**L** M H	L M **H**
Quality	Functionality	L **M** H	L M **H**
	Reliability	L **M** H	L M **H**
	Performance	**L** M H	L M **H**
	Security	**L** M H	L M **H**

Table 3: Technical Comparison between Legacy Supermarket Management System and the Re-Engineered System

	Legacy Supermarket Management System	Re-Engineered Supermarket Management System
Platform	Windows	Windows & linux
Application	Windows based	Web Based
Language	VB	C#
Front end	VB	Asp.net
Backend	Sqlserver	Sqlexpress
Database Re-engineering	All tables are dependent upon the reference constraint which results in changing the condition difficult as it can lead to the removal of old data that do not match the condition.	The constraints are implemented through c# application codes which results in data independency and thereby eases alteration of condition by just changing the code with no effect on the tables and database.
Execution time	The legacy system with VB as its front end takes more execution time, because it includes the execution of both the control and its backend code.	The asp.net with active-x controls takes less time to execute as it requires the execution of only the control and not the back end code.
Coding	Visual Basic programming language which supports the structured programming paradigm is used and hence leads to rewriting the codes since VB does not support grouping control.	C# which supports OOP is used as the programming language and thus enables the programmer to reuse the predefined objects and classes in an easy manner without having any need to write them again.
Making dll (dynamic link library) library	Not supported in VB	The ASP.Net supports the development of of dll files that can be used easily when needed, reducing time and effort.
Number of tables	7	10
Authentication levels	None	Tow levels, Administrator and Employee
Number of reports	3	9
Stock management	Not available	Available
Customer Feedback	Not available	Available

CONCLUSIONS

The new era is marked by the rapid changes in the computer industry that introduces new hardware and software, making older systems suffer from market competition and difficulty in maintenance. From the tabulated values in table (2) and table(3), we found that the FUSION Software re-engineering process model reduces risk level, cost, cyclomatic complexity, and effort. The use of this re-engineering approach provides a way to increase testability, portability, modifiability, functionality, reliability, performance, and security of legacy software. These results prove that the FUSION model is an effective process to perform the re-engineering of legacy software.

ACKNOWLEDGMENTS

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 their great support.

REFERENCES

1. Ahmed S. Abbas et al, “Proposed Software Re-engineering Process That Combine Traditional Software Reengineering Process With Spiral Model”, International Journal of Advanced Research in Computer Science, Volume 4, No. 2, Jan-Feb 2013.

2. L Manzella. Mutafelija.: Concept of re-engineering Life Cycle, ICSI Second International Conference On System Integration. IEEE. 1992.
3. 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.
4. Shekhar Singh, Significant role of COTS to design Software Reengineering Patterns, International Conference on Software Engineering and Applications(ICSEA),2009.
5. 15. Pooley R., Stevens P., Systems Reengineering Patterns, CSG internal report, 1998.
6. Brodie, Michael L., and Stonebraker, Michael, “Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach”, Morgan-Kaufman Publishers, 1995.
7. Alessandro Bianchi et al, " Iterative Reengineering of Legacy Functions", IEEE International Conference on ISBN: 07695118 Year: 2001 Pages: 632-641 Provider: IEEE Publisher: IEEE.
8. 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.
9. Sneed, H., "Planning the Reengineering of Legacy Systems," *IEEE Software,* January 1995, pp. 24–25.
10. SEI (Software Engineering Institute), http://www.sei.cmu.edu/,