International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:2682
Vol.4,No.7, July 2015 DOI:10.15693/ijaist/2015.v4i7.64-67

Software Re-Engineering a New Framework Using Reusable Assets Maintenance

Dr.V.R.ELANGOVAN
Assistant Professor/Department of Computer Applications
A.M.Jain College, Chennai, India

Abstract: As everything becomes changes, the computerized world more and more software have to be changes and evolved. As the software grows vertically and horizontally, many problems have to be emerged in terms of software development and maintenance. Basically the development phase handling one to three years after three years to fifteen years maintance phase is going on. Software Re-Engineering means reorganizing and modifying existing project or software systems and services to make them more maintainable software structure as a composition of components is helpful to enable software maintance, as well as to improve software maintance, more efficiency and quality, this concept of reuse has been implemented very much to reduce the implementation cost and for efficient maintenance.
In this research paper, implement this software re-usability concept, with efficient methodology and frame work. The main aim of the proposed methodology to develop a software component that act as a reuse Asset, which can be used to develop various software with less cost and easily to maintance, customer satisfaction.

Key words – Re-usability, Re-engineering, SDLC, Framework, software development, Re-usability asset maintance.

1. INTRODUCTION

1.1 Software Engineering.

Software engineering as the technological discipline, with systematic production and maintance of software products that are developed and modified, cost estimates on time. Software engineering used to specify, design, implement, validate maintance software products within the time and budget constraints established for the project.
Software engineering project plan contains the lifecycle model to be used software engineering can be a difficult to work that needs a set of software to develop software. Developing software is not at all easy task. The consists of number of phases such as.
Different phases in software Development
Initial study of the project/product
Requirement/analysis
Design
Coding
Testing/Verification
Implementation
Maintenance

All this phase performed various activities in developing the software so all phases in equal importance. The maintance phase, because the maintance basically the development phase handling the project/product one to three years. Three to fifteen years maintance team will handling the project, the maintain software lifelong by managing based on the customer requirements.

1.2 Software Re-Engineering

Software Re-engineering the main purpose is the technological changes or inevitable in the software company. The software application or system, program or product need to be flexible enough to adapt to the changing requirements and also full fill the customer dements the process of transformation of a product with the vision of fulfilling new business requirement along with the existing business requirements is called software re-engineering.

1.3 Factors of Software Re-Engineering

COST FACTORS
PERFORMANCE OF PROGRAM(OR)PRODUCT
USABILITY
DATA STORAGE REQUIREMENTS
NEW TECHNOLOGY TRENDS

Fig-1: Factors of Software Re-Engineering

1.3.1 COST FACTORS
Maintenance cost of the old software system is higher

1.3.2 PERFORMANCE OF PROGRAM (OR) PRODUCT
Higher performance Demands from the customer

1.3.3 USABILITY
Easy and robust handling of software and flexible system designs

1.3.4 DATA STORAGE REQUIREMENTS
Introduction of new data type and increasing quantity of the data

1.3.5 NEW TECHNOLOGY TRENDS

To develop the new technologies to update the software accordingly

2. Related Work

The author Mohamed, Hind Alamin, and Hany H. Ammar.Discussed in the paper [1] survey on the current state of the art in documenting the architectures of existing software systems using reverse engineering techniques. We compared existing methods based on their findings and limitations. The main observation is that existing methods are focused on the developer's concerns and viewpoints as the main stakeholder. We outlined several open issues for further research to develop alternative approaches of reverse engineering for documenting the architectures for development and evolution.

The author Ryan, C. (2000).Discussed in the paper[2] Software Re-Engineering. Software maintenance is a highly important, yet often neglected, part of the software life cycle, and has been widely recognized as one of the more serious contributors to the current demand for programmers. Few systems remain static after delivery, as most are often subject to numerous calls for change, for everything from changing customer needs to porting the system to different operating systems or even different versions of the same operating system.

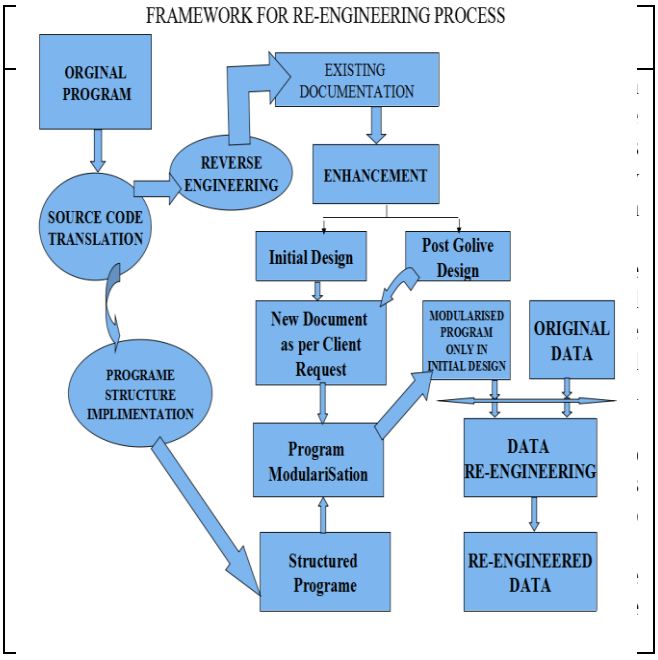
The author MA Babar, L Zhu, R Jeffery Discussed in the paper [3] Software architecture evaluation has been proposed as a means to achieve quality attributes such as maintainability and reliability in a system. The objective of the evaluation is to assess whether or not the architecture lead to the desired quality attributes. Recently, there has been a number of evaluation methods proposed. There is, however, little consensus on the technical and nontechnical issues that a method should comprehensively address and which of the existing methods is most suitable for a particular issue. We present a set of commonly known but informally described features of an evaluation method and organize them within a framework that should offer guidance on the choice of the most appropriate method for an evaluation exercise. We use this framework to characterize eight SA evaluation methods.

The author Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., & Selby, R. Discussed in the paper [4] the stakeholder win–win approach to software engineering education. The key stakeholders we are trying to simultaneously satisfy are the students; the industry recipients of our graduates; the software engineering community as parties interested in improved practices; and ourselves as instructors and teaching assistants. In order to satisfy the objectives or win conditions of these stakeholders, we have formed a strategic alliance with the USC Libraries to have software engineering student team's work with Library clients to define, develop, and transition USC digital library applications into operational use. This adds another set of key stakeholders: the Library clients of our class projects. This paper summarizes our experience in developing, conducting, and iterating the course. It concludes by evaluating the degree to which we have been able to meet the stakeholder-determined course objectives.

The author Morisio, M., Ezran, M., & Tully, C. Discussed in the paper [5] Key factors are derived from empirical evidence of reuse practices, as emerged from a survey of projects for the introduction of reuse in European companies: 24 such projects performed from 1994 to 1997 were analyzed using structured interviews. The projects were undertaken in both large and small companies, working in a variety of business domains, and using both object-oriented and procedural development approaches. Most of them produce software with high commonality between applications, and have at least reasonably mature processes. Despite that apparent potential for success, around one-third of the projects failed. Three main causes of failure were not introducing reuse-specific processes, not modifying non reuse processes, and not considering human factors. The root cause was a lack of commitment by top management, or non awareness of the importance of those factors, often coupled with the belief that using the object-oriented approach or setting up a repository seamlessly is all that is necessary to achieve success in reuse. Conversely, successes were achieved when, given a potential for reuse because of commonality among applications, management committed to introducing reuse processes, modifying non reuse processes, and addressing human factors.

3. Frame work for Reengineering process

In this research paper implemented new framework, the proposed model Inilized two things one is product improvement and another one is product Enhancement. The first thing that we can to do is to analysis whether the task of the software developer ends with the deployment of the software or still continued to the maintance phase if it is continued to maintance phase, then the project is said to maintenance project or enhancement project. In this kind of projects, the programmer has to minor modifications as per the requirement of the client. To to this, the programmer has to work from analysis phase to deployment phase.



re-engineering system.
ii) Program modularized based on new document created because maintain the software by modifying the software as per the client requirement.
iii) In this proposed framework to implement the new empowering technology and implement the post go-live design method.

Table-2

INTIAL DESIGN	POST GOLIVE DESIGN
This is to be done at the initial stage of the project based on the analysis and requirements report.	Post Golive design is done by the design team analyzing what are the modification in the design phase may be occurred. After the project delivery and what are tools to be required to do that.

This table shows the reusable software and our methodology meet all these characteristics and proves that our proposed methodology is the better one. In the comparative to existing one in our new methodology this will reduce our manual power development time and cost, continues improvement also the reusability technology can be implemented in all phase.

CONCLUSIONS

The success of the software Re-engineering the existing software part or all of a system restructuring or rewriting without changing its functionality when some sub modules of a larger system require frequent maintenance.

i) Reengineering easy to maintain to maintains team, and should be re-documented, may also be restructured a Reengineering easy to maintain to maintain team, and should be redocumented, may also be restructured the reengineering system. Through our proposed methodology work done with the software quality and reliability can be increased with less time and cost, this will reduce our manual power continues improvement also ,which save the production cost improving the innovative technology from the existing one reuse is never risk free.

REFERENCES:

[1] Hugo Brunelière, Jordi Cabot, Grégoire Dupé, Frédéric Madiot, MoDisco: A model driven reverse engineering framework, Information and Software Technology, 2014, 56, 8, 1012.
[2] Ryan, C. (2000). Software Re-Engineering. In Automatic Re-engineering of Software UsingGenetic Programming (pp. 17-30). Springer US.
[3] Babar, M. A., Zhu, L., & Jeffery, R. (2004). A framework for classifying and comparing software architecture evaluation methods. In Software Engineering Conference, 2004. Proceedings. 2004 Australian (pp. 309-318). IEEE.

[4] Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., & Selby, R. (1995). Cost models for future software life cycle processes: COCOMO 2.0. Annals of software engineering, 1(1), 57-94.

[5] Morisio, M., Ezran, M., & Tully, C. (2002). Success and failure factors in software reuse. Software Engineering, IEEE Transactions on, 28(4), 340-357

[6] Campbell, John D., Andrew KS Jardine, and Joel McGlynn, eds. Asset management excellence: optimizing equipment life-cycle decisions. CRC Press, 2011.

[7] Chung, S., An, J. B. C., & Davalos, S. (2007, January). Service-oriented software reengineering: SoSR. In System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on (pp. 172c-172c). IEEE.

[8] Jarzabek, Stan. "Domain model-driven software reengineering and maintenance." Journal of Systems and Software 20.1 (1993): 37-51.

[9] Ahrens, Judith D., Noah Prywes, and Evan Lock. "Software process reengineering: Toward a new generation of case technology." Journal of Systems and Software 30.1 (1995): 71-84.

[10] Mulcahy, James J. "Analyzing Software Repository Data to Synthesize and Visualize." In Conference on, June, vol. 231, p. 240

Authors Profile

Dr.V.R.Elangovan received the Doctoral degree in computer science from 2014 Madurai Kamarajar University; I have currently working in Asst professor A.M.Jain College in Chennai