International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2019): 7.583

Software Re-engineering Process Model

Ajit Kumar

Ajit Kumar B.R.Ambedkar Bihar University, Muzaffarpur, India
ajit.phd2013[at]gmail.com

Abstract: ***Software Re-engineering process is the examination and alteration of a system to reconstitute it in a new form. The principle of Re-Engineering when applied to the software development process is called software re-engineering. It affects positively at software cost, quality, service to the customer and speed of delivery. In Software Re-engineering, we are improving the software to make it more efficient and effective. Software Re-engineering is an economical process for software development and quality enhancement of the product. The process enables us to identify the useless consumption of deployed resources and the constraints that are restricting the development process so that the development process can be made easier and cost-effective and maintainable.***

Keywords: Software Re-Engineering, software development. Deployed resources, process model

1. Introduction

Software Re-engineering process takes, it requires involvement of money, and absorbs resources. Re-engineering process is not accomplished in few months but it may take more times. Re-engineering of an information system is the activity that will absorb information technology resources for many years. This is the basic need of an organization for Re-engineering activity. Re-engineering is a rebuilding activity and a user can understand the activity of an information system.

Re-engineering is a rebuilding activity and a user can understand the activity of an information system. The software Re-engineering process model considers the followings

Inventory Analysis –This step of software Re-engineering process model contains the following information

Name of application
Creation date
Number of substantive changes required
Total effort to make these changes
Date of last substantive changes
Effort applied to make last changes
System or system in which it was installed
Application which it interfaces
Name of database which was used by the application
Errors reported over the 12 or 18 months
Number of users
Number of machine on which it is installed
Complexity of program architecture, code and documentation
Quality of documentation
Overall maintainability or scale value of maintainability
Projected life of application
Projected number of changes over next 24 or 36 months
Cost of maintenance (per year)
Cost of business value(per year)
Business criticality

Figure 1: Points in Inventory Analysis in Software Engineering Process Model

The 20 information listed above should be collected for every active application. By sorting this information according to business criticality, longevity, current maintainability and other important criteria, candidates for Re-engineering appears. It is important for an application that the inventory table should be revised regularly.

3. Document Restructuring – The documentation is trademark of many old systems. I need to do the following for restructuring of documents.

Suggestion1. Creating documentation is time consuming process. I need to create a relatively static program. Because a program may contains tens, hundreds or thousands lines of code (KLOC).

Suggestion2. Documentation must be updated .But I have limited resources. If any changes are made to an application then it must be reflected in original documentation.

Suggestion3. The system is critical for business so it must be fully documented.

3.1 Reverse Engineering - The term “reverse engineering” has its origin in “world of hardware”. A company disassembles a competitive hardware product in an effort to understand its competitor’s design and manufacturing “secrets”. Reverse engineering derives one or more design and manufacturing specification for a product by examining actual specimen of the software product.

3.2 Reverse engineering is a process of design secrets. Reverse engineering tools extract data, architectural and procedural design information from existing system or software.

3.3 Code Restructuring – The most common type of Re-engineering is code restructuring. Some old existing systems have relatively solid program architecture but individual modules Ire coded in a way that makes them difficult to understand, test and maintain. In such cases, the code within the suspected modules can be restructured. To avoid the code related error, code restructuring are used. It uses the concept of structured programming. The resultant restructured code is reviewed and tested to ensure that no anomalies have been introduced. Internal code documentation is also updated.

3.4 Data Restructuring –It is a full scale Re-engineering

activity. I generally discuss about current data architecture of a system. In most cases data restructuring uses reverse engineering activity. Since data architecture has a strong influence on program architecture and algorithm that populate it, changes to the data will invariably result in either architectural or code level changes.

3.5 Forward Engineering – The concept of forward engineering uses an automated "Re-engineering engine". The old program would be fed into the engine. The Computer Aided Software Engineering (CASE) Tools may also apply in this field. CASE Tools are limited to specific domains. Forward engineering, also called renovation or reclamation, which extracts recovered design related information from existing software and information to reconstitute the existing in an effort to improve its overall quality. In most cases, it reimplements the function of the existing system and also adds new function to improve overall performance.

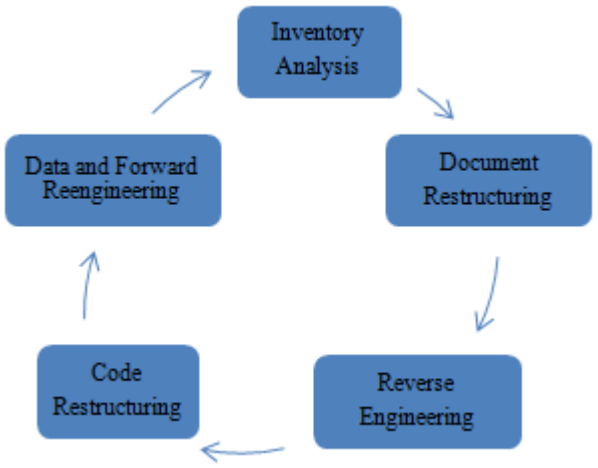

Figure 2: Software Re-engineering Process Model

4. The need of software Re-engineering: Software re-engineering is an economical process for software development and quality enhancement of the product. This process enables us to identify the useless consumption of deployed resources and the constraints that are restricting the development process so that the development process could be made easier and cost-effective (time, financial, direct advantage, optimize the code, indirect benefits, etc.) and maintainable. The software Re-engineering is necessary for having-

a) Boost up productivity: Software Re-engineering increase productivity by optimizing the code and database so that processing gets faster.
b) Processes in continuity: The functionality of older software product can be still used while the testing or development of software.
c) Improvement opportunity: Meanwhile the process of software Re-engineering , not only software qualities, features and functionality but also your skills are refined, new ideas hit in your mind. This makes the developers mind accustomed to capturing new opportunities so that more and more new features can be developed.
d) Reduction in risks: Instead of developing the software product from scratch or from the beginning stage here developers develop the product from its existing stage to enhance some specific features that are brought in concern by stakeholders or its users. Such kind of practice reduces the chances of fault fallibility.
e) Saves time: As we stated above here that the product is developed from the existing stage rather than the beginning stage so the time consumes in software engineering is lesser.
f) Optimization: This process refines the system features, functionalities and reduces the complexity of the product by consistent optimization as maximum as possible.

5. Re-Engineering cost factors:

- The quality of the software to be re-engineered.
- The tool support availability for engineering.
- The extent of the data conversion which is required.
- The availability of expert staff for Re-engineering.

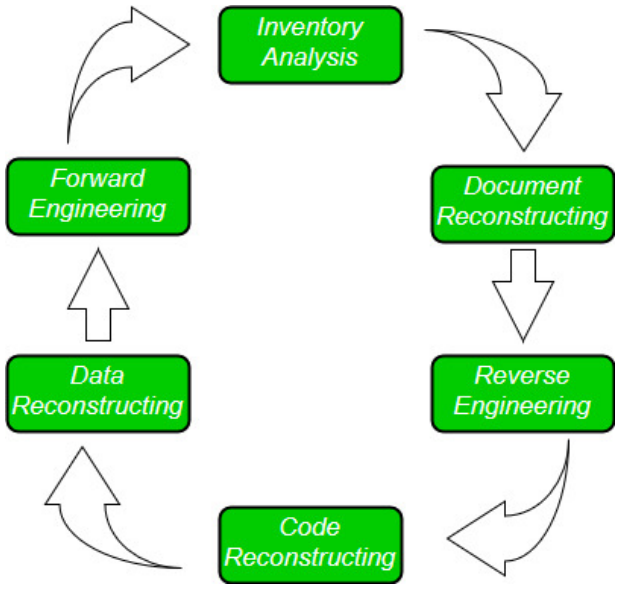

Figure 3: Re-engineering cost factor

References

[1] [Cas98] Eduardo Casais. Re-engineering object-oriented legacy systems. Journal of Object-Oriented Programming, 10(8):45–52, January 1998.
[2] [CC90] Elliot J. Chikofsky and James H. Cross, II. Reverse engineering and design recovery: taxonomy. IEEE Software, pages 13–17, January 1990.
[3] [CGK98] Yih-Farn Chen, Emden R. Gansner, and Eleftherios Koutsofios. A C++ data model supporting reachability analysis and dead code detection. IEEE Transactions on Software Engineering, 24(9):682–693, September 1998.
[4] [Ciu99] Oliver Ciupke. Automatic detection of design problems in object-oriented Re-engineering . In Proceedings of TOOLS 30 (USA), pages 18–32, 1999.
[5] [Com94] CDIF Technical Committee. CDIF framework for modelling and extensibility. Technical Report EIA/IS-107, Electronic Industries Association, January 1994. See http:/www.cdif.org/.
[6] [DD99] Stéphane Ducasse and Serge Demeyer, editors. The FAMOOS Object-Oriented Re-engineering Handbook. University of Berne, October 1999.

[7] [DDHL96] H. Dicky, C. Dony, M. Huchard, and T. Libourel. On automatic class insertion with overloading. In Proceedings of OOPSLA'96, pages

Author Profile

Ajit kumar received MCA degree from IGNOU and doing PhD. Degrees from B.R.Ambedkar Bihar University Muzaffarpur. Currently he is working as a faculty at S.R.K.G.College Sitamarhi in Department of BCA.