International Journal of Advanced Research and Development
ISSN: 2455-4030, Impact Factor: RJIF 5.24
www.newresearchjournal.com/advanced
Volume 1; Issue 4; April 2016; Page No. 70-72

Necessitate of reengineering strategy through case tools for free SDLC

Karthik S

Assistant Professor Department of Computer Science PSG College of Arts & Science, Coimbatore, TamilNadu, India.

Abstract
Software Engineering is a process used towards identify the accuracy completeness and quality of the well developed software (Software Re-engineering). It includes a set of activities, conducted with the intent of finding bugs or errors in software so it corrected before product released to the client or end users. Software testing is an activity to check whether actual results match the expected results and to ensure that software system is error free or defect free.
Software testing is more than just error detection and software re-engineering introducing for generating error free software. Testing software is operating the software under controlled conditions, to verify that it behaves as specified, finding bugs/errors and validate that what has been specified is what the user's requirements. Software testing is more than just error detection and software re-engineering introducing for generating error free software. Testing software is operating the software under controlled conditions, to verify that it behaves as specified, finding bugs/errors and validate that what has been specified is what the user's requirements. Product Re-engineering focuses on alteration of an alive product, occasionally through reverse engineering. The objective of re-engineering a product is towards optimizing its performance by means of adding new functionalities as well as taking benefit of emerging technologies.

Keywords: System re-engineering, Data mining, Clustering, Software re-engineering, Hybrid-reengineering, CASE tools

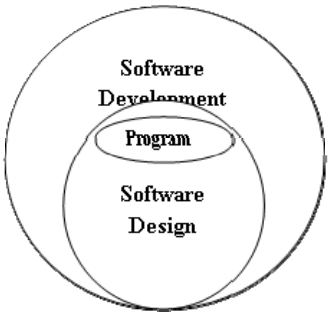
Introduction
The world is running forward to achieve endless limits by innovative thoughts every day. In this scenario, the entire science, engineering and the arts are developing from their end node to subsequent steps. This particular thesis is the entire concerning the next step in addition to re-structuring of last node or end. Here in this world, nothing is innovative one. In the meantime, in this similar approach need towards follow in dissimilar point of hypothesis throughout Re-engineering.
Organized information within the form of operating systems, utilities, programs as well as applications, enable computers in the direction of work. Software consists of cautiously organized instructions as well as software source code written through programmers or else developers in any of a variety of special computer-programming languages.
A major system upgrade will totally replace an existing system with new technology, including a new language, a new software engineering environment and a new operating system. It did not provide for retraining the existing engineers to learn new technologies and new approaches or adding engineers with the required skills. As a result, the organization will need to have a long-term maintenance contract with an outside user that will tie the organization to the end user until either the existing workforce retires or a newer work force takes over.

2. Software Development Paradigm
Paradigm is a software engineering paradigm where the entire the engineering concepts pertaining towards the development of software are applied. It includes a variety of researches as well as requirement gathering, which assists the software product towards build. It consists of:

- Requirement gathering
- Software design
- Programming (coding)

Data mining as well known, as Knowledge Discovery in Databases (KDD) it is the process of extracting potentially supportive information as of raw data. A software engine can scan huge amounts of data as well as automatically report interesting patterns without requiring human intervention. Few other knowledge discovery technologies are Statistical Analysis, online analytical processing (OLAP), Ad-hoc queries, and data-visualization. These technologies, data mining do not require a human being to ask specific questions (Artificial Intelligence).
Clustering is a typical unsubstantiated learning technique for grouping similar data points. A clustering algorithm assigns a huge number of data points headed for a smaller number of groups so as to data points in the similar cluster share the similar properties while, in dissimilar groups, they are dissimilar.

2.1. System Re-engineering
Restructuring or else rewriting part otherwise the entire of a system without changing its functionality
Applicable while a few subsystems of a larger system require frequent maintenance
Reengineering involves putting in the effort towards make it easier to maintain
The reengineered system might also be restructured as well as should be re-documented



Structure for Programming Paradigm

2.2. Software Re-Engineering

The process of software of application (web or windows) with the vision of full filling new Application Requirements along with the Existing Application Requirements is called Software Re-Engineering.

Software reengineering is reorganizing as well as modifying existing software systems towards create them additional maintainable. Reengineering involves adding effort towards making them easier to maintain, the system restructured as well as re documented. The purpose of reengineering is headed for modernize the existing system over a newer one. Aging as well as unreliable system components, while the systems turn out to be outages frequently. Changes towards business processes turn out to be too complex, difficult, and or costly to implement. Solutions are less expensive than legacy maintenance. Advantages of reengineering reduce risk at reduced cost. The risk identification is an art. The risk identification is additional significant intended for effective risk assessment, risk analysis, as well as management. Monitoring methods explained meant for the categorized risks. It will assist a reengineering system in the direction of an ease of maintenance as well as cost benefit by means of reduced risk at reduced cost.

2.3. Hybrid-Reengineering

Software re-engineering, a recent research area includes reverse engineering & forward engineering while Hybrid re-engineering incorporates both the engineering processes 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.

3. Reengineering With the Business Process Change

Reengineering is generally discussed as "business process change". Such change imposes new requirements on systems. It 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. 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. The legacy system is replaced by a new system with the same or improved functionality.

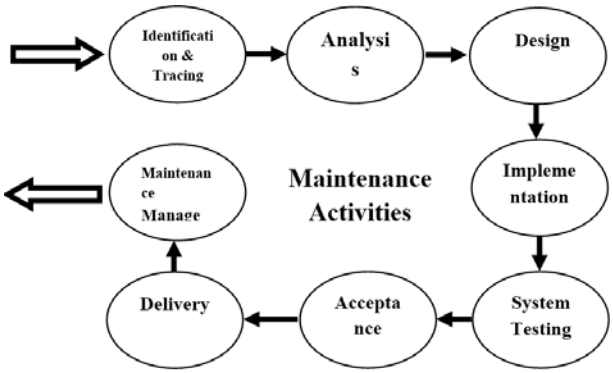
4. Case Tools

When a new system is installed, the implementation integrates a figure of related as well as dissimilar tasks. The process encompasses efficiently organized in addition to it is intended for this cause that CASE tools are developed. Through the assist of CASE, the installation process can be automated along with coordinated in the developed moreover adopted system life cycle. CASE is the software engineering tools so as to permit collaborative software development in addition to maintenance.

4.1. Need of Case Tools

The software development procedure is expensive in addition to, the projects turn out to be more complex in nature; the project implementations turn into more demanding as well as expensive. That is why software developers forever looking meant for such CASE tools so as to assist in a lot of dissimilar ways throughout the dissimilar development stages of software, so as to understand the software in addition to prepare a good end product so as to efficiently fulfil the user requirements.

CASE tools endow with the integrated homogenous environment intended for the development of complex projects. These tools provide computerized setting towards software developers in the direction of analyze a problem in addition to design its system model. The CASE tools also offer the environment meant for monitoring as well as controlling projects such that team leaders are able towards managing the complex projects.



4.2. Life-Cycle Based Case Tools

This dimension categorizes CASE Tools on the source of the actions they support in the information systems life cycle. They can be categorized as Upper or else Lower CASE tools.

- Upper CASE Tools support strategic planning in addition to construction of concept-level products as well as ignoring the design aspect. They support traditional diagrammatic languages for instance ER diagrams, Data flow diagram, Structure charts, Decision Trees, Decision tables, etc.
- Lower CASE Tools focus on the back end activities of the software life cycle, for instance physical designs, debugging, constructions, testing's, component integrations, maintenances, reengineering as well as reverse engineering.

4.3. Risks and Associated Controls

Common CASE risks as well as allied controls comprise:

- **Inadequate standardization:** Linking CASE tools as of dissimilar vendors (design tool commencing Company ABC, programming tool from Company XYZ) might be difficult if the products do not use standardized code structures in addition to data classifications. File formats

can be converted, however frequently not economically. Controls comprise using tools as of the similar vendor or else using tools based on standard protocols in addition to insisting on demonstrated compatibility.

- **Unrealistic expectations:** Organizations frequently implement CASE technologies towards reducing development costs. Implementing CASE strategies typically engages high start-up costs. Normally, management have to be willing towards accepting a long term payback period.
- **Slow implementation:** Implementing CASE technologies is able to involve an important change as of traditional development environments. Characteristically, organizations have not use CASE tools the primary time on crucial projects or else projects by means of short deadlines since the lengthy training process.
- **Weak repository controls:** Failure towards sufficiently control access towards CASE repositories might consequence in security breaches or else damage towards the work documents, system designs or else code modules stored in the repository. Controls comprise protective the repositories by means of appropriate access, version in addition to backup controls.

5. Conclusion

As a result of far above the ground of business environment pace changing, RAD demonstrate itself at dramatically shorten the SDLC. Though, as of the view of a system requirements analysis, RAD intensively combines the entire types of system analysis methodologies into one-step process through using nowadays high-powered computer technology. RAD a lot adopts JAD and prototyping as the two main ways towards gathering the requirements, so the intrinsic worth and disadvantages of these two methods also can be found in RAD. In sequence to adapt to the OOP, a whole set of OOA needs structuring tools, such as use case diagram, class diagram and so on, were developed, which are methodically dissimilar to the traditional requirements structuring tools, such as DFD, structured English, etc. Consequently far, whether the OOA is improved than the traditional analysis methods otherwise not still a question mark. Nevertheless, traditional requirements analysis techniques were enthused by as well as founded on, structured programming concepts. These days, in a programming world so as to be increasingly turning towards object orientation; such traditional techniques seemed outdated as well as had to be replaced.

6. References

1. El Hamdouni1 A-E, Djamel Seriai1 A, Huchard M. Component-based Architecture Recovery from Object Oriented Systems via Relational Concept Analysis, LIRMM/CNRS, Montpellier 2 University, France, 2010.
2. Li A-P, Wang Z-h, Duan L-G, Li X–P. Study and application of legacy system reengineering based on component reuse. Journal of Applied Sciences. 2013; 13:8.
3. Adolph W Stephen. Cash Cow in the Tar Pit: Reengineering a Legacy System. IEEE Software, 1996.
4. Aiken PH, Muntz A, Richards R. DoD legacy systems: reverse engineering data requirements, Communications of the ACM, 1994; 37(5).
5. Aman Jatain, Deepti Gaur. Reengineering Techniques for Object Oriented Legacy Systems, International Journal of Software Engineering and Its Applications. 2015; 9:1.
6. Brodie Michael L, Stonebraker Michael. Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach", Morgan-Kaufman Publishers, 1995.
7. Pooley R, Stevens P, Software Engineering Patterns, SEBPC workshop, 2008.
8. Pooley R, Stevens P. Systems Reengineering Patterns, CSG internal report, 1998.
9. Shekhar Singh, Significant role of COTS to design Software Reengineering Patterns, International Conference on Software Engineering and Applications (ICSEA), 2009.
10. Sommerville, Ian, Software Engineering, AWL, 2000.
11. Staffan, Sandell D. Reengineering and Reengineering Patterns, the Department for Computer Science and Engineering, Mälardalens Högskola Västerås, 2002.