about: can be used as second paper

International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 6 (2018) pp. 3648-3654
© Research India Publications. http://www.ripublication.com

EESRM: An Effective Approach to Improve the Performance of Software Re-Engineering

A. Cathreen Graciamary [1], Dr. M.Chidambaram [2]

[1] *Research Scholar, Bharathiyar University, Coimbatore, India.*
[2] *Assistant Professor, Rejah Serfoji Govt College, Thanjavur, India.*

Abstract

Nowadays, development of software is described as the continuous advancement of software products. These software products are updated frequently for the duration of their usage. In several cases, software systems are developed by altering the architecture or by including new attributes because of new business plans or technologies. Re-engineering of Computer software is an economical and ideal way to provide much needed boost to user's current software system. In existing software Re-engineering techniques, there is no efficient technique to improve and reduce the existing software. In this research work, our proposed system presents Efficient and Enhanced Software Re-engineering Mechanism to resolve the existing software re-engineering issues. In our proposed system, the data is converted into byte and stored into separate file. Then, perform classification with that particular data set. Comparing with existing system, our proposed system performance is high because the access time of byte size of data is very lower than other data types. Moreover, our proposed mechanism reduces the complexities in software re-engineering. Finally, we conduct set of experiments to prove our proposed mechanism is better than other approaches.

Keywords: Software Engineering, Software Re-engineering, Software Quality, Restructuring.

INTRODUCTION

The process of modifying the present computer software to get used to change new software development is known as Software Re-Engineering. But basically, re-examine and analyzing its contents, transform the necessary contents (or the entire computer software as required), break it apart, also its getting the completed product that is: software, and then finally put it exact back together another time [1]. As this type of activity is the real process of Application, then it can also be known as Reverse Engineering. When the process of re-engineering for computer software occurs then the technique of organization is also being improved for being in the line within enhanced software technology. The software for the re-engineered computer might be simple for the comparison with the older methods like knowledge and ideal methods are sufficient in market. There are huge risks generated with the development of project for computer software. One of the major risks is known as the problem identification.

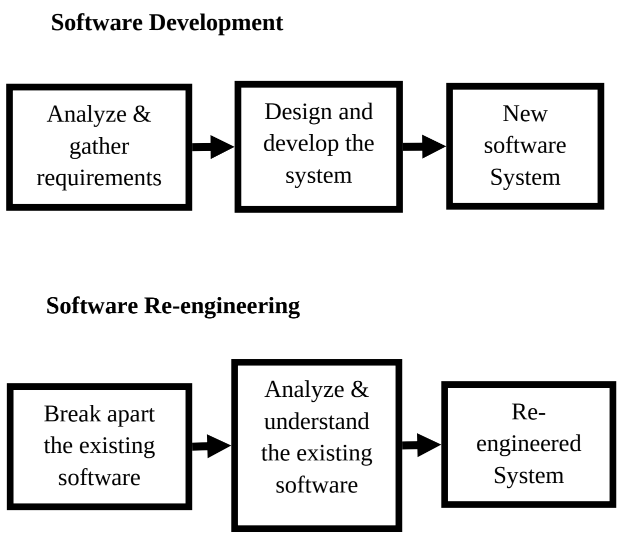

Software Development
Analyze & gather requirements
Design and develop the system
New software System
Software Re-engineering
Break apart the existing software
Analyze & understand the existing software
Re-engineered System

Figure 1: Software development V/s Software re-engineering

Fig 1 illustrates the general structure of software development and Software Re-engineering [2].

Solitary issues are producing some phases throughout the life cycle of software that could enlarge it into a huge problem because of the flawed software for generation. Re-engineering of Computer software might assist not simply support to stick any issues, which was produced during its generation but it steadily reduce the leaning in re-creation for a similar issue related when construction of a new software system [3].

Re-engineering of Software has developed into significant sub discipline inside CS (Computer Science) from past ten years. Huge firms gradually automate the critical and important jobs this makes their company extremely reliant on Information Systems. But in several cases, these software systems were sustained for a couple of years, these are legacy software systems that are significant for the firm, but they are often complex to maintain and understand. In terms of budget and time, the re-development of these software systems from scrape is not a better plan. Therefore, other alternative is software re-engineering.

In software re-engineering of a software system, one can study about software re-engineering history, techniques and tools of reengineering, re-engineering phases. Re-engineering system's cost and its experiences have to be observed to design an absolute re-engineering scheme. Various research works have different significance for software re-engineering.

Software Re-engineering is the process of analysis examination and modification of existing software application to re-constitute it in a new structure, and the succeeding completion of the new structure [4]. The re-engineering concept is the modification and examination of an existing system to rebuild it in a new structure and the succeeding execution of the target form [5].

Software Re-engineering is defined as "Re-engineering concerns the design examination and existing legacy system implementation and relating several methods and mechanisms to reshape and redesign that software application into optimistically more suitable and better software application [6]. Re-engineering has three main parts such as: (I) Forward Engineering (FE) [5], (II) Reverse Engineering (RE) [4] and (III) architecture transformation. RE is a major part of the software re-engineering life cycle.

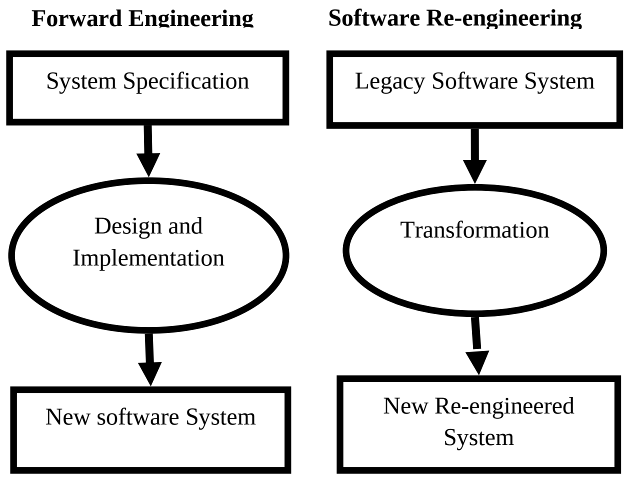


Figure 2: Forward Engineering and Software Reengineering

Fig 2 show the general structure of forward engineering and software re-engineering.

At present, software re-engineering of object oriented system (OOS) is the future research field since software experts are now managing huge volume of industrial program, implemented with the help of object oriented methods of the late eight's and early ninety's. While managing these software systems, several problems may takes place and the software experts have to efficiently prevent and fulfill the changing requirements of the customers [7]. Though the concept of object oriented is moderately new, there are a lot of OOSs. The concept of object oriented paradigm was established by several software ventures but currently with the survival of millions LOC (line of code) in OOSs, software re-engineering of these software applications is a speeding research domain. Currently, object oriented paradigm adopter handling the problems related to transformation of these OOSs into the more re-usable framework and mechanism. So, methodologies or tools maintenance is needed to handle the huge and badly documented OOP (Object Oriented Programs). The software re-engineering of OOS was gaining special treatment to reduce the growth costs and to assist the software protection [8].

Taxonomy of Software Re-Engineering

This sub section provides the taxonomy and scope of the software re-engineering domain with several key terms as follows:

1. FE (Forward engineering)
2. RE (Reverse engineering)
3. Re-documentation
4. Reverse design or Design recovery
5. Program comprehension or Program understanding
6. Restructuring
7. Recode
8. Redesign
9. Re-specify

Forward engineering is the conventional procedure of moving from sophisticated logical and abstractions, execution not-dependent designs to the system's physical completion.

Reverse engineering is the analyzing process of software system to (i) make out the software's interrelationships and their mechanism and (ii) generate system representations of the current system in a different structure or in a higher abstraction level.

Re-documentation is the revision or creation of a semantically correspondent illustration inside the similar comparative level of abstraction. It is also the oldest and simplest reverse engineering form and should be measured to be an un-invasive, weak restructuring form [9].

Reverse design or Design recovery re-creates abstractions of the design from a mixture of existing design documentation, code, personal experience, application domains and general information about the problem.

A term **Program comprehension or Program understanding** is related to reverse engineering process. Program comprehension implies that comprehension starts with the program while RE may start at an executable and binary structure of the system or at sophisticated descriptions of the system design [11].

Restructuring is the process of transforming the data from one representation structure to a different at very similar comparative level of abstraction, while protecting the existing software system's semantics and functionality [10].

Recode process consists of altering the implementation characteristics of the program. Control flow restructuring and language translation are the program-level changes [12].

Re-design is the process of modifying the characteristics of design. Feasible changes contains improving an algorithm, restructuring design architecture, changing a data of a system model as incorporated in a DB (database) or in data structures [14].

Re-specify process consists of modifying the characteristics of the requirements. This kind of change may refer to altering the existing requirements structure only [13].

As a result, the existing researches discusses about the re-engineering of software using different kinds of techniques. In order to overcome the existing drawbacks in software re-engineering, our proposed system presents a novel approach namely Efficient and Enhanced Software Re-engineering mechanism. In this mechanism, without changing the implemented technology it increases the performance of existing software application. The details of our proposed methodology are described in section III.

The rest of the paper is organized as follows. Section II which overviews the related work. Then we provide the detailed description of our proposed scheme in Section III. Section IV gives the results and discussions of our proposed scheme. Finally, Section V concludes the whole research work.

RELATED WORK

Research work [15] comprise software re-engineering in business procedure change the alternations within the time period in a firm; however, this condition provides several similar issues in which a software system implemented in one firm and it should be utilized in another firm. Research work [13] states that expert in software re-engineering are very fewer than the experts in development and design and the majority of the software engineers don't have large amount of research knowledge in Re-engineering area [13]. The issues with existing software system are present all over the globe. The author in [17] describes an existing software system as individual, which considerably resists modification and development to meet continuously changing & new business needs without considering the well-known technology to design the software system. Here, the existing software system is replaced with a new software system with similar or enhanced functionality.

Nowadays, significant consideration was devoted to the software reengineering phenomenon [18] [19]. In the earlier periods, research studies in the reengineering area mainly focuses on the various reengineering frameworks development, but only a few research studies discovered the risk factors in software re-engineering process. Risks of re-engineering and their power on quality of software cause the efforts of re-engineering to be unsuccessful. For successful re-engineering development, the required re-engineering risks were organized. Research work [22] discusses a plan, for software re-engineering to develop the cost-effective large software re-engineering features. On the other hand, management, economic, and quality features of software plays a vital part in the successful completion of software re-engineering efforts. Research work [21] reviews the outcome of thirteen software re-engineering projects accomplished over the last decade. Analysis illustrates that the re-engineering projects have considerably lesser risk factor than development projects - twenty five percents as opposed to fifty seven percents. This variation could be measured by the following terms such as cost overruns and project completion rates.

Moreover, re-engineering selection is also essential in considering the other aspects such as resource utilization, user satisfaction performance improvement and quality goals, etc. Research work [20] recommends 2 major reasons for software re-engineering failure as political risk and functionality risk, correspondingly. Although, there are other serious risks like process risk, technical risk, architecture risk, risk related to stakeholders and development environment risk are also the reasons for the efforts of software re-engineering to be unsuccessful.

Research work [23] developed a metrics structure to assess the difficulty of existing software to support product outsourcing. The structure consists of 2 legacy system dimensions such as source code and documentation. Moreover, several other dimensions of an existing software system like managerial system, and technical dimensions take part in a vital role in development process of software system. Research work [24] explains a gradual reengineering process of the technical mechanism of existing software system. This mechanism allows the existing system to be steadily emptied into the re-engineered software system without needing to duplicate the legacy system. Research work [25] proposed a novel model for legacy software systems namely A Dual-Spiral Re-engineering Model, this executes cyclic approach. The major work flow in this model needs 2 software systems (target system and legacy system) to work jointly; then shift the functionality of the system from the existing software to the new software gradually, based on the spiral model. For the period of whole re-engineering procedure, the active functionality of the traditional software is in decremented model and the functionality of the novel target software is in an incremental model.

PROPOSED WORK

Overview

Efficient and Enhanced Software Re-engineering Mechanism is a software re-engineering process which is used to transform a traditional software system to a new software system. The figure 1 shows the overall structure of our proposed re-engineering mechanism. Initially, our system analyze the drawbacks of existing terms in terms of accessing time of the software application and storage size of the software application. After that, analyze the solution to overcome the drawback of existing system within the implemented technology. Then check the feasibility for improving the performance of the existing system. Then check the feasibility to solve the existing problem. Finally, compare the result of modified product with existing product in terms of access time and memory usage. **Fig 3** shows the overall Proposed System.

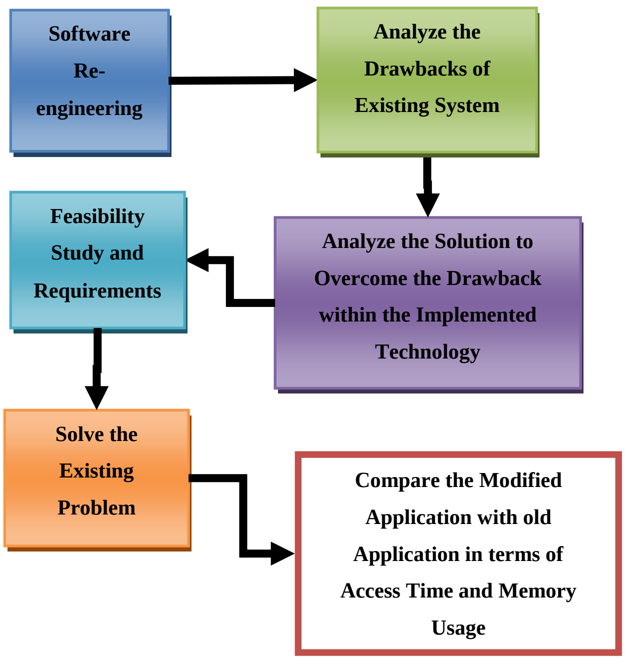

Software Re-engineering
Analyze the Drawbacks of Existing System
Analyze the Solution to Overcome the Drawback within the Implemented Technology
Feasibility Study and Requirements
Solve the Existing Problem
Compare the Modified Application with old Application in terms of Access Time and Memory Usage

Figure 3: Architecture of Proposed System

Working Methodology

Our proposed Framework consists of three major stages. Such as

1. Analyze the Concept of Existing System
2. Analyze the Solution with the Existing Technology
3. Feasibility Study and Solve the Problem
4. Compare the Result of Modified Application with Existing Application

Analyze the Concept of Existing System

The preliminary process of our Efficient and Enhanced Software Re-engineering Mechanism is analyzing the concept of existing system. In this stage, the developer analyzes the concept of existing system and its system requirements like front end, back end, and hardware specifications. In brief, it initially re-specify the current requirements or needs of the customer. The causes for selected software system failure were identified within the 1st stage, several software systems do not have any kind of security or some other drawbacks; there is feeble fault management process that doesn't cover the entire probable fault cases. After analyzing the abovementioned things, the developer must know the performance of the existing software application like processing time and memory usage, etc.

Analyze the Solution with the Existing Technology

This stage describes the details about the analysis of finding the solution to overcome the drawback within the implemented technology and re-testing process. Before finding the solution, identifying the existing problem or current condition of the existing technology is very crucial process of software re-engineering. The current condition of the traditional software re-engineering consists of operability and maintainability. A software metrics set must be chosen to help in detecting the quality issues with the existing software system and the application prioritization that are applicants for software re-engineering based on their business value and technical quality. In addition, converting the current technology of source code into new technology is also a complicated process; alternatively, the expert finds the optimal solution with current technology which reduces the cost and memory consumption.

Feasibility Study and Solve the Problem

This stage describes the details about the feasibility study of the proposed solution. After finding the optimal solution for re-engineering the existing software application without changing the current technology, check the feasibility of the proposed solution which is also a huge problem. For checking the feasibility, consider the following things such as to increase quality of software, to improve the efficiency of maintenance process, and enhance the business value. While establishing the prospects, the user should be expressed in an assessable way – such as cost reduction in performance improvement, sustaining engineering and operations reduction, etc. After that, the reengineering costs should be compared to the predictable cost savings and to increase the value of current software application.

Compare the Result of Modified Application with Existing Application

This stage is the final stage of our Efficient and Enhanced Software Re-engineering mechanism. Based on the result of previous software re-engineering stages, the implementation of a software application or system should be performed. In implementation, a particular part might be replaced with the best one which completely relies on the prior three stages of our Efficient and Enhanced Software Re-engineering mechanism. Finally, it compares the performance of modified software application with existing software application with the attributes like processing time and memory usage.

RESULTS AND DISCUSSION

Experimental setup

To evaluate the performance of our Efficient and Enhanced Software Re-engineering mechanism, we conducted a series of experiments on three different types of dataset such as Breast cancer dataset, Diabetes and Heart Disease dataset, the detailed description of the datasets are given in **table 1**. In these experiments, we implemented and evaluated the proposed techniques in following configuration: 2 GB RAM, Processer speed 2.90 GHz, Intel i3(R), CPU G2020, and Operating system -Windows 7, and Front End -JAVA.

Detail of Dataset

For Diabetes,

Attribute 1: Number of times pregnant

Attribute 2: Plasma glucose concentration a 2 hours in an oral glucose tolerance test

Attribute 3: Diastolic blood pressure (mm Hg)

Attribute 4: Triceps skin fold thickness (mm)

Attribute 5: 2-Hour serum insulin (mu U/ml)

Attribute 6: Body mass index (weight in kg/(height in m)^2)

Attribute 7: Diabetes pedigree function

Attribute 8: Age (years)

Attribute 9: Class variable (0 or 1)

For Heart disease,

Attribute 1: age

Attribute 2: sex

Attribute 3: chest pain type (4 values)

Attribute 4: resting blood pressure

Attribute 5: serum cholesterol in mg/dl

Attribute 6: fasting blood sugar > 120 mg/dl

Attribute 7: resting electrocardiographic results (values 0, 1, 2)

Attribute 8: maximum heart rate achieved

Attribute 9: exercise induced angina

Attribute 10: old peak = ST depression induced by exercise relative to rest

Attribute 11: the slope of the peak exercise ST segment

Attribute 12: number of major vessels (0-3) colored by fluoroscopy

Attribute 13: thal: 3 = normal; 6 = fixed defect; 7 = reversible defect

Attribute 14: Absence (1) or presence (2) of heart disease

For Brest Cancer,

Attribute 1: Clump_Thickness integer [1, 10]

Attribute 2: Cell_Size_Uniformity integer [1, 10]

Attribute 3: Cell_Shape_Uniformity integer [1, 10]

Attribute 4: Marginal_Adhesion integer [1, 10]

Attribute 5: Single_Epi_Cell_Size integer [1, 10]

Attribute 6: Bare_Nuclei integer [1, 10]

Attribute 7: Bland_Chromatin integer [1, 10]

Attribute 8: Normal_Nucleoli integer [1, 10]

Attribute 9: Mitoses integer [1, 10]

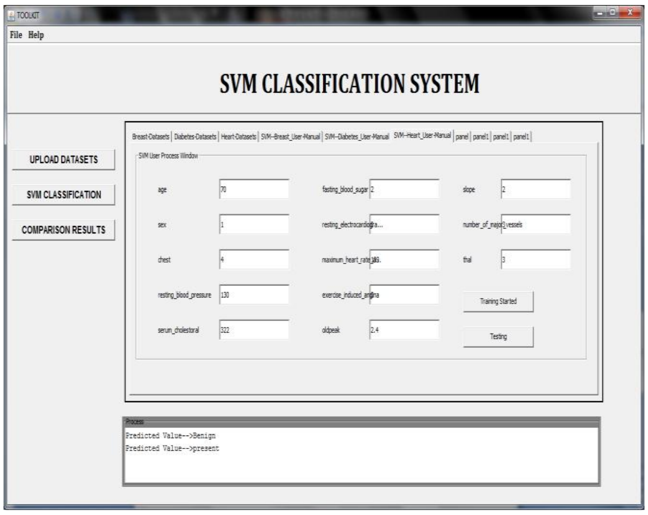
Attribute Class {benign, malignant}

Performance Evaluation

To evalaute the performance of our Efficient and Enhanced Software Re-engineering Mechanism, we have formed one medical diagnosis & prediction system for three different diseases like breast cancer, diabeties and heart disease patients. Here, we consider two medical diagnosis and predication applications such as

- SVM Classification System
- Modified SVM Classification System

For estimating the performance of our Re-engineering system, first obtain the input dataset. For SVM Classification System, no datatype conversion should performed, the expert use MySQL as back end for storing the datasets. For Modified SVM Classification System, dataset is converted into bytes and store it in the file instead of database. In data processing, the expert can convert the byte data into string and then precede the classification process.

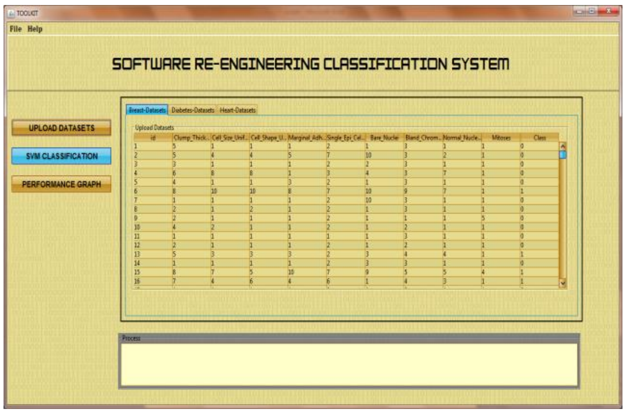

TOOLKIT
File Help
SVM CLASSIFICATION SYSTEM
Breast-Datasets | Diabetes-Datasets | Heart-Datasets | SVM-Breast_User-Manual | SVM-Diabetes_User-Manual | SVM-Heart_User-Manual | panel | panel1 | panel1 | panel1
SVM User Process Window
UPLOAD DATASETS
SVM CLASSIFICATION
COMPARISON RESULTS
Training Started
Testing
Process

age	70	fasting_blood_sugar	2	slope	2
sex	1	resting_electrocardiogra...		number_of_major_vessels	3
chest	4	maximum_heart_rate	109.	thal	3
resting_blood_pressure	130	exercise_induced_angina			
serum_cholestoral	322	oldpeak	2.4		

```
Predicted Value-->Benign
Predicted Value-->present
```

Figure 4. SVM Classification System

Fig 4 illustrates a medical diagnosis and prediction system by using Support Vector Machine Classification Algorithm for predicting several diseases like heart disease, diabetes and breast cancer. The classification operation is performed without changing the data type of the datasets.



id	Clump_Thick...	Cell_Size_Unif...	Cell_Shape_U...	Marginal_Adh...	Single_Epi_Cel...	Bare_Nuclei	Bland_Chrom...	Normal_Nucle...	Mitoses	Class
1	5	1	1	1	2	1	3	1	1	0
2	5	4	4	5	7	10	3	2	1	0
3	3	1	1	1	2	2	3	1	1	0
4	6	8	8	1	3	4	3	7	1	0
5	4	1	1	3	2	1	3	1	1	0
6	8	10	10	8	7	10	9	7	1	1
7	1	1	1	1	2	10	3	1	1	0
8	2	1	2	1	2	1	3	1	1	0
9	2	1	1	1	2	1	1	1	5	0
10	4	2	1	1	2	1	2	1	1	0
11	1	1	1	1	1	1	3	1	1	0
12	2	1	1	1	2	1	2	1	1	0
13	5	3	3	3	2	3	4	4	1	1
14	1	1	1	1	2	3	3	1	1	0
15	8	7	5	10	7	9	5	5	4	1
16	7	4	6	4	6	1	4	3	1	1

Figure 5. Modified-SVM Classification System

Fig 5 illustrates the proposed medical diagnosis and prediction system named Modified-Support Vector Machine Classification system for predicting the different types of diseases like heart disease, diabetes and breast cancer.

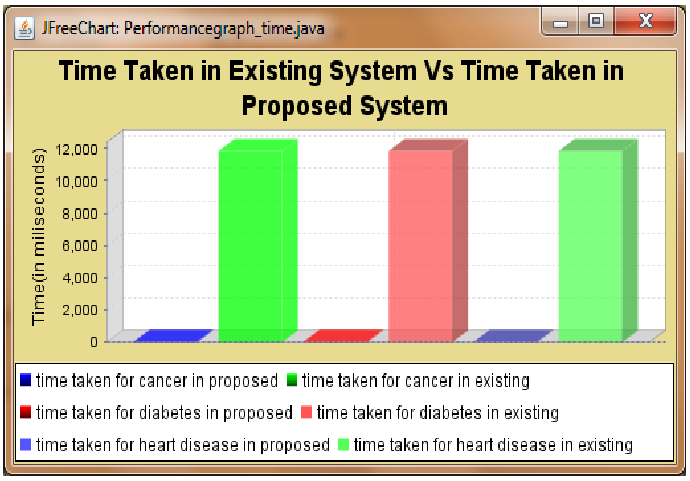


Figure 6. Accessing Time Comparison of Proposed System and Existing System

Fig 6 The performance graph illustrates the processing time comparison between the proposed system (Modified-SVM Classification System) and existing system (SVM Classification System). Here, the graph shows our proposed mechanism takes very less time for prediction process of three types of dataset like cancer, diabetes and heart disease.

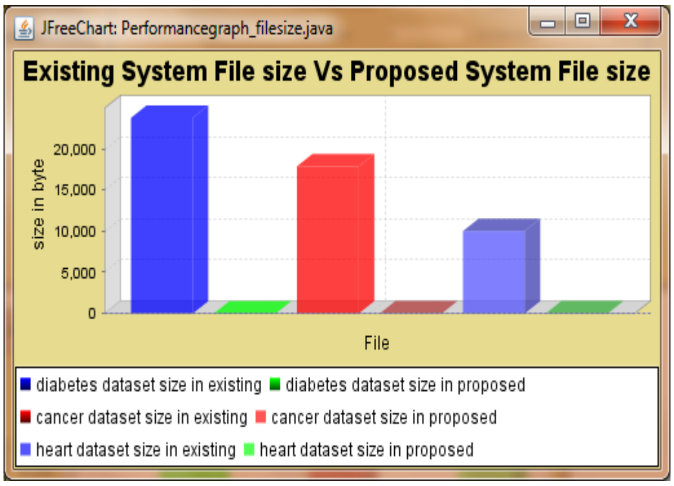


Figure 7. Memory Usage Comparison of Proposed System and Existing System

Fig 7, performance graph illustrates the memory usage comparison between the existing system (SVM Classification System) and proposed system (Modified-SVM Classification System). Here, the graph shows our Modified- SVM Classification System which utilize minimum memory space for prediction process for three types of dataset like cancer, diabetes and heart disease.

Table 1. Datasets Details

S.NO	DATASET NAME	ATTRIBUTES	CLASS
1.	Diabetes	08	02
2.	Heart Disease	13	02
3.	Breast-Cancer	09	02

CONCLUSION

Currently, a huge amount of changes are rapidly involved in software and hardware due to the fast development of computer industry. Re-engineering of software provides minimized risk level in software development. Actually, new software application development is more risky process. So, a novel Software Re-engineering mechanism is required to convert the existing system into a new system without changing the implemented technology of existing system. This Efficient and Enhanced Software Re-engineering mechanism is proposed to reduce the memory usage and processing time. Finally, our proposed mechanism improves the QOS with minimum development efforts and increases the efficiency, reliability of software application.

REFERENCES

[1] Anquetil, Nicolas, and Jannik Laval. "Legacy software restructuring: Analyzing a concrete case." In SoftwareMaintenance and Reengineering (CSMR), 2011 15th European Conference on, pp. 279-286. IEEE, 2011.

[2] M. A. Serrano, D. L. Carver and C. Montes de Oca, “Reengineering legacy systems for distributed environments”, Journal of System Software, Elsevier, vol. 64, (2002), pp. 37–55, (2002).

[3] A.-P. Li, Z.-h. Wang, L.-G. Duan and X.–P. Li, “Study and application of legacy system reengineering based on component reuse”, Journal of Applied Sciences, vol. 13, no. 8, (2013), pp. 1233-1238.

[4] L. H. Rosenberg, “Software Re-engineering”, Lawrence E. Hyatt Manager, Software Assurance Technology Center System Reliability and Safety Office Goddard Space Flight Center, 1997, NASA 301-286-7475.

[5] D. Staffan and D. Sandell, “Reengineering and Reengineering Patterns”, The Department for Computer Science and Engineering, Mälardalens Högskola 2002-02-24 Västerås.

[6] E. J. Chikofsky and J. H. Cross II, “Reverse engineering and design recovery: A taxonomy”, Index Technology Corp. and Northeastern University and Auburn university, (1990).

[7] C. Szyperski, “Component Software; Beyond Object-Oriented Programming”, Addison-Wesley, (1998).

[8] B. Caprile, A. Potrich and P. Tonella, “Reverse Engineering of Object Oriented Code”, Geneva (CH), (1999).

[9] Tahvildari, L., Kontogiannis, K. On the role of design patterns in quality-driven re-engineering. in Proceedings of the IEEE 6th European Conference on Software Maintenance and Re-engineering (CSMR). 2002. Hungary.

[10] Harry.M.Sneed, “Economics of Software re-engineering”, Journal of Software Maintenance, Vol.3, 1991, p.163

[11] M.Solvin, and S. Malik. “Re-engineering to reduce system maintenance: A case study”,Software Engineering, pp.14-24,2011.

[12] S. Ducasse, T.G.ı., and J.-M. Favre, Modeling software evolution by treating history as a first class entity, in on Software Evolution Through Transformation 2004. p. 71–82.

[13] J. Ransom, I.S., I. Warren. A Method for Assessing Legacy Systems for Evolution. in Proceedings of the 2nd Euromicro Conference on Software Maintenance and Re-engineering (CSMR'98). 1998.

[14] Moghaddas, Y., & Rashidi, H. (2009). A novel approach for replacing legacy systems, Journal of Applied Sciences, 9(22), 4086–4090.

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

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

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

[18] Ransom, J., Somerville, I., Warren, I., “A method for assessing legacy systems for evolution”, in Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering, 1998, ISBN: 0-8186-8421-6, INSPEC Accession Number: 5884288,PP 128 – 134.

[19] Paul Briden, “Software Re-engineering process,” Tessella Support Services PLC Technical report, Issue V2.R1.M1, 2000.

[20] Eric K. Clemons Michael C. Row Matt E. Thatcher, “An Integrative Framework for Identifying and Managing Risks Associated With Large Scale Reengineering Efforts,” in 1995, Proceedings of the 28th Annual Hawaii International Conference on System Sciences - 1995 pp. 960-969.

[21] Harry M. Sneed, “Risks Involved in Reengineering Projects,” in WCRE: Proceedings of the Sixth Working Conference on Reverse Engineering-1999, IEEE Computer Society pp.204.

[22] Peter H. Feiler, “Reengineering: An Engineering Problem,” Technical Report Software Engineering Institute Carnegie Mellon university Pittsburgh Pennsylvania 15213, CMU/SEI-93-SR-5, 1993.

[23] Cristiane S. Ramos, Káthia M. Oliveira, Nicolas Anquetil,” Legacy Software Evaluation Model for Outsourced Maintainer”, published in CSMR '04 Proceedings of the Eighth Euromicro Working Conference on Software Maintenance and Reengineering (CSMR'04), IEEE Computer Society Washington, DC, USA ©2004 table of contents ISBN:0-7695-2107-X

[24] Alessandro Bianchi, Danilo Caivano,Vittorio Marengo, Giuseppe Visaggio,” Iterative Reengineering of Legacy Functions”, 17th IEEE International Conference on Software Maintenance (ICSM'01), Florence, Italy, ISBN: 0-7695-1189-9,November 07-November 09.

[25] 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.