Reengineering Framework to Enhance the Performance of Existing Software

Jaswinder Singh[1]
Department of Computer Application
IK Gujral Punjab Technical University
Kapurthala, Punjab, India

Kanwalvir Singh Dhindsa[2]
Department of Computer Science and Engineering
BBSB Engineering College
Fatehgarh Sahib, Punjab, India

Jaiteg Singh[3]
Department of Computer Applications
Chitkara University
Rajpura,Punjab,India

***Abstract*—Term reengineering refers to improve the quality of the system. Continues maintenance and aging degrade the performance of the software system. Right approach and methodology must be adapted to perform reengineering. With lack of right approach and methodology, reengineering itself will be costly and time-consuming. For the process of reengineering main concerns include when to reengineer, how to estimate cost, the right approach for reengineering, and how to validate software enhancement. This research paper proposed a framework to identify the need for reengineering, to estimate the cost of reengineering, and to validate software quality improvement. Research work used the agile methodology to perform tasks of reengineering. Reengineering needs are identified using prediction based decision tree approach. Reengineering is applied using the agile Scrum methodology. Cost estimation is done using story point estimation. Performance analyses are done using complexity measures analysis of the internal design metrics and mean time to execute metric. The research used various automated tools like CKJM ver1.9, Rapid Miner studio ver7.1, and Net beans7.3 framework.**

***Keywords*—reengineering; maintenance; decision tree; agile methodology; scrum**

I. Introduction

Maintenance is one of the most critical phases of software development. Continues maintenance degrades the software quality and increases the maintenance cost. The software reengineering plays a vital role to improve the quality of software. Reengineering is required to upgrade the existing system and to reduce the maintenance cost. Many researchers [1, 2] proposed a framework for reengineering identification and reengineering cost estimation. But these frameworks are not able to handle the ever-changing behavior of customer needs and requirements. These existing frameworks lack the flexibility to adopt the changes and as well as to estimate cost. Earlier approaches are based upon conventional engineering methods. Since a few decades, we have also seen changes in software development approach, especially with the use of agility in software development. So need is to use a comprehensive approach to provide a new framework for software reengineering that is flexible as well as an interactive model to adopt the customer requirements and able to perform the cost estimations. This research work proposed a framework to identify the need for reengineering, estimate the cost of reengineering, uses an agile approach, reduce the maintenance cost of the reengineered system and finally evaluate the performance of reengineering system. Proposed research work provides a vision for developers to quickly identify reengineering needs and able to apply to reengineer in a people-centric environment using agile. Research work in this paper organized under different sections. Related work discussed in the literature review section. Section 3 describes the research methodology used in this paper. Another section identifies whether the software is required to be reengineered or maintained. Reengineering agile model and estimations discussed in Sections 5 and 6. Performance evaluation is given in the last section.

II. Literature Review

The existence of a reengineering approach is not new. It has been observed that due to continuing changes in the existing software, software quality deteriorates [3] and reengineering must be performed to adapt the changing requirements of end-user. Researchers identified [4] the importance of reengineering and stated the importance of information technology in software reengineering. Reengineering performs preventive maintenance for the software system [5]. Reengineering includes three important subtasks named reverse engineering, restructuring or alteration, and forward engineering. Reengineering tasks are shown in Fig. 1 [6]. Researcher [7] also identified various benefits like better software quality, fewer maintenance efforts, and ease of software testing and a better understanding of the software. Sneed observed the impact of reengineering over maintenance [8]. Researchers [9] proposed a cost model for reengineering using the conventional approach. Agile methodology has proven to be a successful approach to software development for the last few years [10]. Agile is integrated with the field of reengineering by many researchers. The researcher proposed N-Process model [11]. N-process model is N-shaped reengineering structure to perform various tasks of reengineering. Tasks are mapped in N shaped structure. Other work gives the idea of service-oriented software reengineering [12]. Service-oriented computing paradigms applied to enhance the legacy systems. Work is also done to provide prototypes at the initial stages of reengineering [13]. Researchers also worked on aspect-oriented reengineering [14]. In aspect-oriented reengineering, reengineering work is validated by applying various object-oriented metrics, and tasks were performed in short iterations of agile.

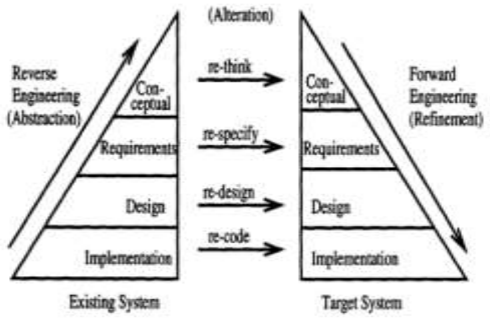

Fig. 1. Software Reengineering [6].

III. Research Methodology

The case study includes twenty open sources, Java-based software systems. The complexity of the Java-based system is measured using Chidamber and Kemerer metric popularly known as CK Metric [15] of object-oriented software. Six basic metric sets of CK metric suit include Depth of the Inheritance Tree (DIT), Number of Children (NOC), Response for a Class (RFC), Lack of Cohesion of Methods (LCOM), Weighted Methods per Class (WMC) and Coupling between Object Classes (CBO). CKJM tool is used to measure the basic set of CK metric. Using CK metric, internal design complexity of the software system can be determined. To identify the need for reengineering, prediction based decision tree approach [16] is used for the software systems. Once the software got categorized for reengineering or maintenance requirements [17], agile development approach is used to get the software reengineered and also to estimate the cost of reengineering. Performance of the reengineered system is evaluated by comparing the design complexity of reengineered and old software. Classes having complex design are the candidate for reengineering.

IV. Identify the Need for Reengineering

The decision among maintenance and reengineering is made using prediction based decision tree approach. Data set consist of twenty software systems divided into two parts. Each part is a mix of varying lines of codes and complexity. Research work considered fifteen software projects as a training data set, and five projects as model data set. Attributes of the training data set will be applied to predict model data sets. For implementing the predictions using a decision tree approach, an average of internal design complexity and size of software systems act as two main metrics. Table I shows Java-based software systems considered under the training data set. Software belongs to different size and having different average internal design complexity. Internal design complexity is measured using a basic set of CK metric suit.

Five projects are considered under the model data set. Table II shows various software systems for model data set. Training data set will be applied to the mode data set to predict reengineering and maintenance requirements.

The process of Applying and executing a decision tree using rapid minor tool is as followed.

- Import training data set having fifteen Java-based software projects.
- Roles are used to selecting attributes. First role operator is used to choosing a category attribute. Average complexity and size are chosen as two parameters.
- The second role is used to skip project names from the analysis part.
- Predictions are made using a decision tree with Decision Tree operator.
- Training data set will be input to the Decision Tree operator.
- The classification model is the output of Decision Tree that will be used for decision making
- Model data set is imported using a retrieve operator. New role is applied to the data set setting parameter ‘Category’ which is required to be predicted.
- There are two outputs of Apply model. One output is the prediction of attributes applied to model data using training data, and other is training data itself.
- The complete design is presented in Fig. 2.
- Finally, decision tree and predictions can be viewed by executing the designed scenario.

TABLE I. Training Data Set Complexity Measure [17]

SrNo	**Software**	**SLOC(Size)**	**Mean Complexity**
1	PongGame Software	713	31.3
2	Software ChessGame	150	29
3	Battle City Software	563	77.2
4	Software Customer Info System	1139	120.3
5	Parser Software	143	13.8
6	Software Scheduling and dispatch	203	82.7
7	Dictionary Software	337	24.7
8	Software ChatServer	284	24.3
9	My Notepad Project	290	2
10	Trigonometric Function Software	634	362.7
11	SoftwareCricketAnalyzer	234	16.7
12	Diary App Software	431	26.3
13	Software TicTacToe	276	12.7
14	FIFO Software	637	75
15	Software BounceBall	160	12.1

TABLE II. Model Data Set Complexity Measure [17]

Sr. No	Software	SLOC(Size)	Mean Complexity
1	E-library Software	323	55
2	Shopping Cart Software	154	24.7
3	Code Level Security Software	201	144.5
4	Point of Sale Software	1082	526.5
5	SmartFileConverter Software	440	39.7

Fig. 2 represents the design interface. Apply Model operator is required to apply a decision tree on the model data set.

Once executed, the decision tree will appear, as shown in Fig. 3. A decision tree is made up of nodes and edges. The root of tree denotes prominent predictor. Thus it is observed that Average complexity is our best predictor of deciding reengineering requirements. It predicts whether or not the Java project requires reengineering. The predicted value for Average complexity comes out to be 25.5. The second node is of size attribute. Thus best predictor at second level is source line of code SLOC (Size). The tree from root to the leaf can be interpreted as if Average complexity >25.5 and SLOC (Size)>176.5 the software undergoes reengineering. Thus except shopping cart software of model data set given in Table II, all other software are the candidate for reengineering.

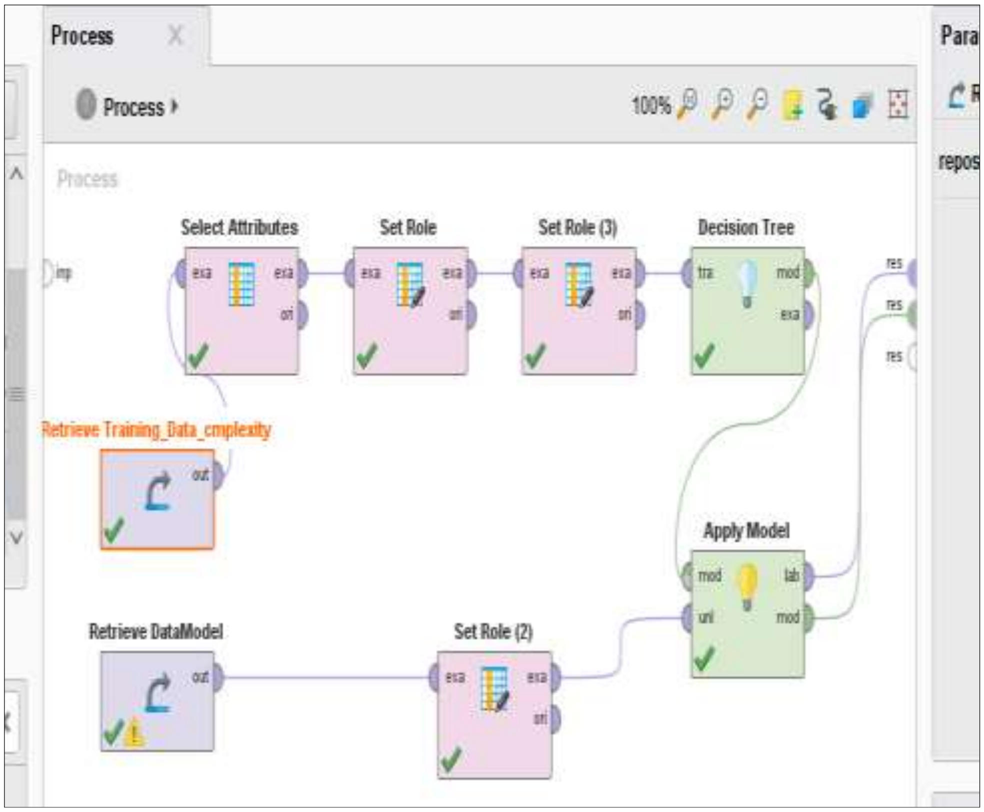

Fig. 2. Decision Tree Modeling in Rapid Miner [18].

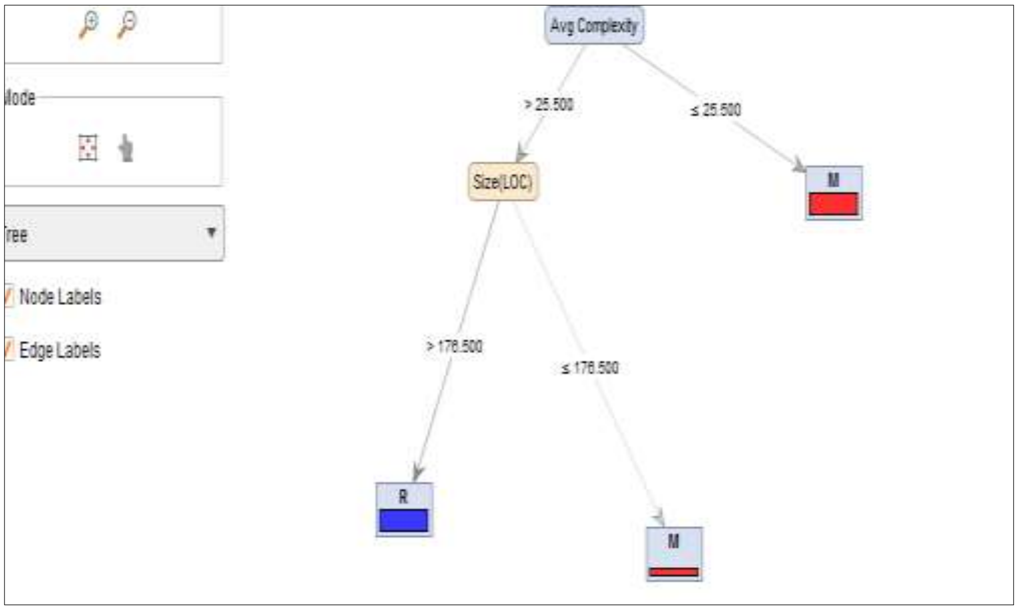

Fig. 3. The Decision Tree Structure for Model Data Set [18].

V. Agile Reengineering Model

Once the software is chosen for performing reengineering, an approach to perform reengineering is required. Among various software development approaches, one of the most popular and acceptable methods for development is agile [19, 20]. Development of software in agile include active participation among various stakeholders of software. Many agile frameworks exist like Scrum, Extreme Programming, Lean programming, United Process, Kanban, FDD (Feature-Driven Development), Crystal, DSDM (Dynamic Systems Development Method). Among these frameworks, Scrum is one of the most useful approaches by IT professionals. In a report of Scrum alliances [21], 89% of agile users used the scrum approach. Major Scrum activities include Scrum planning; Daily Scrum, sprint review, and sprint retrospective shown in Fig. 4. Sprint represents a single iteration in fix time. Many sprints can be used to develop the required product. Requirements are analyzed in terms of user stories, and estimation is performed by assigning story points to each user story. Whole requirements are collected as a product backlog. Requirements of high priority assembled in the sprint backlog. In sprint planning, the work required to perform decided. A product backlog is analyzed, and sprint backlog is prioritized in this phase. The team meets every day to evaluate the progress of the sprint. The team reviews the work and changes required. The team finally discusses goals achieved and if anything went wrong, ways of improvement.

Because of the flexible and interactive approach of software development, it is decided to perform reengineering using agile methodology. The inclusion of reengineering tasks with agile scrum methodology is shown in Fig. 5. Proposed agile reengineering model retains the essence of reengineering and agility. Three tasks of reengineering are performed using an Scrum methodology. All three reengineering tasks are enclosed in one sprint of three-week iteration.

Agile reengineering model works as follows:

- Ensure planning of release of reengineering software, planning of iterations (time allocation for iteration, team members required, etc.), and estimation of cost. All requirements are prioritized in the product backlog.
- Requirements required to implement in one sprint are assigned to the sprint backlog. Planning is done by the Scrum team, including all stakeholders.
- Analysis of Reengineering Requirements in terms of user stories and allocation of story points.
- Execution of sprint with 3-week iteration to accommodate forward, alteration and reverse engineering
- Retrospective action to confirm the implementations of required objectives. After iteration, estimation, and speed of requirement implementations (velocity) is verified.
- Daily planning is performed every day.
- One sprint perform reverse, alterations and forward engineering
- Integration for final complete System.

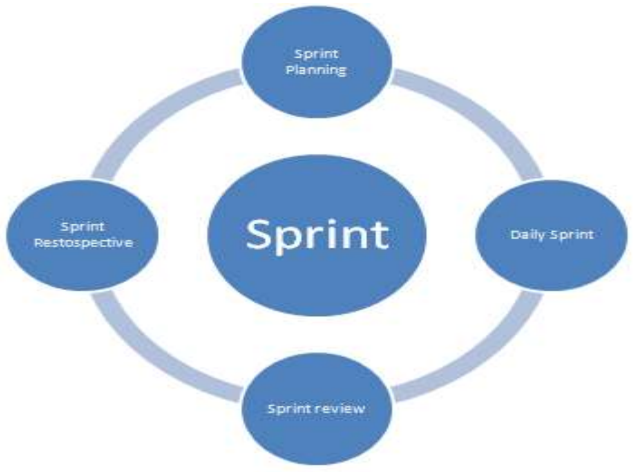

Fig. 4. Scrum Activities.

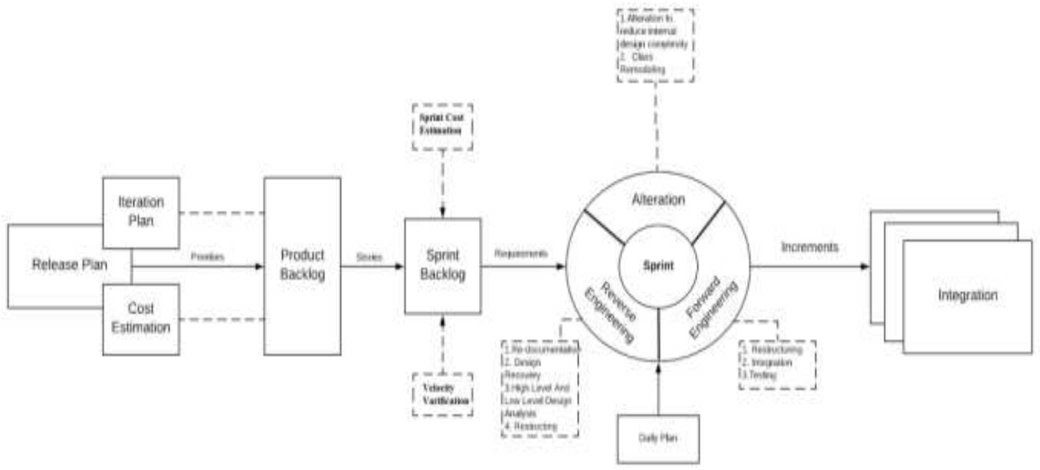

Fig. 5. Agile Reengineering Model.

VI. Estimations

Estimations of effort and cost are essential for any projects. One important aspect of proposed framework is to estimate efforts and cost of existing systems. Proposed work measures effort and cost estimations of reengineering with the help of an agile approach.

A. Efforts Estimations

Reengineering efforts are estimated by assigning story points to the required tasks. Planning poker also called Scrum poker is highly acceptable techniques for assigning story points to reengineering requirements. As stated by Cohen [22], "Planning poker is a proper mix of expert opinion, analogy, and disaggregation techniques which can successfully give quick and reliable estimates.

Benefits of planning poker include

- Scrum team, including Scrum master, product owner, and development team (developer, testers, Analyst) sit together to perform estimations.
- Both high and low estimation points for user story are discussed. Meeting avoids the problem of conflict for the future.
- Work starts when all members are agreed upon the same consensus so commitment for the project increases.

As everyone gets a chance to justify himself and everyone‘s opinion is welcomed, so no chance of dominance of individual arises.

B. Cost Estimations

Reengineering cost is estimated considering the cost of human resources, Time required to complete the tasks, cost of other resources required (hardware, software licensing, etc.).Sprint is planned, and the time of one sprint is estimated. Formulations of Various cost estimations are as follows:

- Let Ann. Sal represents the annual salary of the Scrum team member.
- Acc.Sal denotes accumulated salary, which is the sum of the annual salary and other expenses. For reengineering process, we can include other expenses as half of the annual salary of employee as suggested by Cohen [22] for software development. Character 'i' denotes n number of members of the Scrum team.

$$(Acc.Sal)i \ \forall \ (i = 1,2 \ldots n) = \ (Ann.Sal)i \ \forall \ (i = 1,2 \ldots n) + \left(\frac{1}{2}\right) * \left((Ann.Sal)i \ \forall \ (i = 1,2 \ldots n)\right) \quad (1)$$

- Let K denotes the number of weeks per iterations then salary per iteration (Sal.Iter) is

$$(Sal.Iter)i \ \forall \ (i = 1,2 \ldots n) = \left(\frac{K}{52}\right) * \ (Acc.sal)i \ \forall \ (i = 1,2 \ldots n) \quad (2)$$

- Let P denotes the estimated number of days required for an employee to work on the project then the percentage of time spent by employees will be

$$(Time.Spent)i\forall \ (i = 1,2 \ldots n) = \left(\frac{1}{5*K}\right) * (Pi\forall \ (i = 1,2 \ldots n) * 100) \quad (3)$$

- Accumulated cost per time spent (Acc.Cost.Time.Spent) for each member of scrum team is

$$(Acc.Cost.Time.Spent)i\forall \ (i = 1,2 \ldots n) = \ ((Sal.Iter)i \ * \ (Time.Spent)i))\forall \ (i = 1,2 \ldots n) \quad (4)$$

The initial cost may be estimated in a long time, and estimations can be reviewed after each sprint.

VII. Case Study

As discussed, reengineering is performed using the Scrum methodology. For our case study, software named CodeLevelSecurity from Table II is chosen for reengineering. Table III shows all the classes of this software. Complexity measures for software are determined using a basic set of CK metric.

Three classes are selected for sprint backlog depending upon their usage and importance in the project. Login, IDE, and UserDetail classes have been chosen to perform reengineering. Several reengineering tasks performed in one sprint are discussed in Table IV. Sprint iteration of 3 weeks is estimated for implementing the required reengineering. Story points assigned to Login, IDE, and UserDetails are 2, 8 and 5 respectively. Each task in sprint backlog is estimated on an hourly basis.

TABLE III. Candidate Software for Reengineering [19].

Sr No	**Classes**	**Design Metrics**					
		WMC	*DIT*	*NOC*	*CBO*	*RFC*	*LCOM*
1	Login	12	6	0	9	8	60
2	IDE	17	6	0	17	21	70
3	UserDetail	23	5	0	12	09	183
4	program access report	12	6	0	8	9	62
5	Profile detail'	11	5	0	6	4	25
6	User report	8	6	0	6	7	24
7	Saved program report	14	6	0	10	8	73
8	User maintenance	2	1	0	1	5	1
9	Program update report	12	6	0	9	94	48
10	Main frame	22	6	0	19	9	233
11	program report	8	6	0	6	4	24

TABLE IV. VARIOUS REENGINEERING TASKS PERFORMED IN ONE SPRINT

S. No	Reengineering Tasks	Hours allotted
1. Reverse Engineering		
1.1	Generating Documentation/re-documentation	6
	Design Recovery	
1.2	High-level design analysis	4
1.3	Low-level design analysis	8
1.4	Analysis of restructuring requirements.	12
2. Alterations and forward engineering		
2.1	Classes Remodeling	6
2.2	Design Complexity reduction in classes through Alterations	18
2.3	Performing Unit test	6
2.4	Performing Regression test	12
2.5	Increment Integration	6
2.6	Testing	6
2.7	Retrospective	6
	Total Sprint Time	90 Hrs

A. *Cost Estimation*

Cost is estimated using the equations (1), (2), (3), and (4). Scrum team includes three members named Scrum master, programmer, and tester. Consider annual Salary of Scrum master, programmer and tester as $1,50,000, $60,000 and &60,000 respectively. Accumulated salary (Acc.Sal) as given in equation (1) is calculated as $225000, $90000, $90000 for each employee. For the reengineering process with three-week iteration (putting K=3 in equation (2)) accumulated salary per iteration (Sal.Iter) is approximately $12981, $5192 and $5192 corresponding to all scrum team. Time Estimation is performed for each Scrum team member. The estimated value of P is15 days for Scrum master, 12 days for programmers and eight days for tester then the percentage of time spent by every member as calculated by equation (3).

For scrum Master, estimated days (P) are 15.

By putting the value of P in equation (3),

(1/(5*3)) * (15*100) that is 100 % time.

For programmer, estimated days (P) are 12.

By putting value of P in equation (3),

(1/ (5*3)) * (12*100) that is 80 % time.

For Tester, estimated days (P) are 8.

By putting the value of P in equation (3),

(1/ (5*3)) * (8*100) that is approximately 53% time.

Accumulated cost per time spent as given in equation (4) for Scrum Master is $12981.Similarly, by putting values in equation (4), Accumulated cost per time spent for the programmer is approximate $4154 and for the tester is $2752. So the total cost of project per iteration is $19887.Thus we can estimate the actual cost of reengineering. Still, we can assume the uncertainty factor which can reduce or increase the actual cost of reengineering. As suggested by Cohen [22], the actual cost in an agile environment can be + or – 25% of estimated values.

B. *Evaluating Complexity Reduction and Performance Improvement of Reengineered Software*

Once reengineering is performed, software is analyzed for complexity reduction and performance up gradation. Outcomes of reengineering interpreted in three ways.

- Complexity in terms of the Basic set of CK metrics has been reduced in reengineered classes.
- Reduction of software complexity results in an improvement in maintainability.
- Improvement in the overall mean time to execution (MTTE) of the project, due to CK metric value reduction.

1) Complexity in terms of the basic set of CK metrics reduced in reengineered classes: It has been observed that by applying reengineering tasks, the inherent design complexity of classes measured in terms of CK metrics has been reduced to a reasonable extent, as shown in Table V. For all the three classes of the project, there is a reduction in WMC, CBO, RFC, and LCOM. Due to reengineering, classes are restructured, and alterations are done at the function level. The numbers of functions and dependencies in each class have been reduced. Comparisons of reengineered and old classes are shown in Table V.

TABLE V. CK METRIC COMPARISON BEFORE AND AFTER REENGINEERING

		Design Metrics						
Metrics & Software Classes		*WMC*	*DIT*	*NOC*	*CBO*	*RFC*	*LCOM*	*Total*
Login Class	Reengineered	4	6	0	5	48	2	65
	Before Reengineering	12	6	0	9	78	60	165
IDE Class	Reengineered	4	6	0	12	102	0	124
	Before Reengineering	17	6	0	17	121	60	221
UserDetail Class	Reengineered	6	6	0	4	67	0	83
	Before Reengineering	23	5	0	12	109	183	332

2) Reduction in the software complexity results in an improvement in maintainability of software system: As stated [23], larger the values of CK metric more will be the software complexity, and hence the software will be more error-prone. Total reduction of CK metric values for all the three classes are shown in Fig. 6. The Fig. 6 shows CK metric analysis for both reengineered and existing candidate classes. On the x-axis, there are three classes, and on the y-axis, CK metric complexity is depicted.

Once the software got reengineered, the maintenance cost of the reengineered project will be undoubtedly low. As suggested by Chaudhary and Ugrasen [24], maintenance can be estimated based on story points. Before reengineering, story points for three classes were fifteen, then after reengineering they are reduced to six only. Story points assigned to Login, IDE, and UserDetails are 1, 3 and 2 respectively. That means

- More classes can be accommodated (if required) in one iteration of the Scrum
- Will results in the reduction of the cost
- less time spent to perform changes
- fewer complexity results in less possibility to induce more errors

So the system once reengineered can survive longer and further can adapt changes (undergo maintenance) with less cost and time.

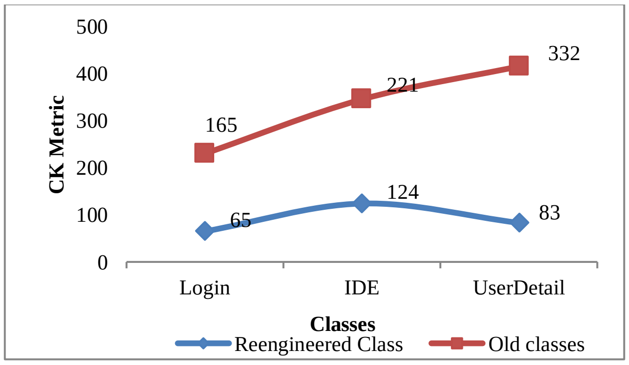

Fig. 6. CK Metric based Maintainability Comparisons.

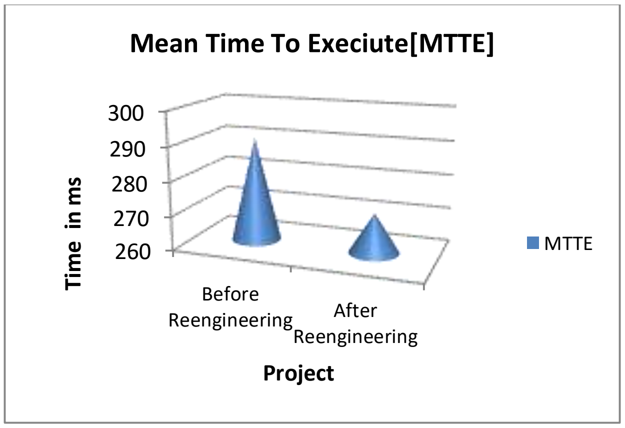

Fig. 7. MTTE Values for Existing and Reengineered Project.

3) Improvement in the overall mean time to execution (MTTE) of the project: Another improvement in the reengineered project is in the meantime to execute (MTTE). For all the three old and reengineered class modules, samples of 35 executions are taken. Net beans7.3 is used with system configuration of i5-4th gen processor, 8GB RAM, HDD 1TB and Java7. MTTE is 290.6 milliseconds for classes of the old project and 271.7 milliseconds for reengineered project classes. MTTE analysis is shown in Fig. 7.

VIII. Results and Discussion

The proposed research work is discussed from Sections IV to VII. Except shopping cart software of model data set given in Table II, all other software is the candidate for reengineering. These software systems are CodeLevelSecurity, PointofSale, E-Library, and SmartFileConvertor. Among these four candidate systems, CodeLevelSecurity is chosen to reengineer. Three classes of the software are reengineered. CK metric suit is used to measure the design complexity of software. Agile Reengineering Model is proposed to perform estimations and to apply reengineering tasks. Two main objectives of the proposed model included:

- To apply agile-reengineering development approach to perform reengineering on the candidate system.
- Performing effort and cost estimations for reengineering.

After performing reengineering, the reengineered system is evaluated for maintainability and performance up gradation. It is validated that the reengineered system performs much better than the existing candidate system. Results for average complexity and MTTE are shown in Table VI.

It is important to note that the candidate software gone through reverse, alteration and forward engineering. After reengineering, the numbers of functions in the three classes are also reduced from thirty three to thirteen. So in place of refactoring, the reengineering process is applied to the software to inculcate requirements of reducing complexity and increasing performance. Not only the complexity of the software is reduced but the performance of the system is also improved.

TABLE VI. Reengineered System Performance Measures

Sr No	Software Type	Average Complexity of three classes	MTTE in milliseconds for complete software
1	Reengineered Software	272	271.7
2	Existing Candidate Software	718	290.6

IX. Conclusion

Proposed work introduced a framework that identifies reengineering requirements for software using prediction based decision tree approach. Agile Reengineering model uses features of the agile development approach with a reengineering approach. Cost estimation is done using story point technique. Complexity and performance analyses are performed using CK metric and MTTE metric. Using agile reengineering approach, reduction in the cost of maintenance and improvement in maintainability observed. After reengineering of classes of software, complexity is reduced to a greater extent. As a result indicates, performing reengineering by using agile methodology is beneficial in terms of implementing requirements, estimating cost, and enhancing performance. Cost estimations are realistic and involve a consensus of all stakeholders. Software complexity in terms of the internal design of software calculated using CK metric. This research can be a benchmark to the software development companies to identify whether software needs maintenance or reengineering. Also, cost estimations can easily be measured for the software to be reengineered.

Further, software complexity can be validated using other software metrics like cyclomatic complexity, reliability, etc. For a generalization of the framework and to make it industry ready more and more software systems can be considered to make an extensive training data set. Larger the training data set, more accurate will be the predictions. The experience of Scrum team will play crucial role to successfully implement Agile Reengineering Model.

ACKNOWLEDGMENT

We like to acknowledge IK Gujral Punjab Technical University for providing resources and support.

REFERENCES

[1] H. M. Sneed, “Estimating the Costs of a Reengineering Project,” Proceeding of 12th Working Conference Reverse Engineering. IEEE CS Press, Pittsburgh, USA , 2005, pp. 111–119.

[2] S. Sood, Software reengineering-A metric set based approach, Himachal Pradesh University.2012.

[3] M.M. Lehman, ”Programs, life cycles, and laws of software evolution,” Proceedings of the IEEE. Vol. 68, No. 9, 1980, pp.1060-107.

[4] M. Hammer and J. Champy, Reengineering the Corporation: A Manifesto for Business Revolution. New York: HarperCollins Publishers, 1993.

[5] S. Ian, Software Engineering, 9th ed., Pearson publication. 2014.

[6] E.J. Byrne, “A conceptual foundation for software re-engineering,” Proceedings of Conference on Software Maintenance 1992, Orlando, FL, USA, 1992, pp. 226-235.

[7] R.S. Arnold, ” Software Restructuring,” Proceeding of IEEE, vol. 77, no. 4, April 1989, pp. 607–617.

[8] H.M. Sneed and A.A. Kaposi, “Study on the effect of reengineering upon software maintainability,” Proceedings. Conference on Software Maintenance 1990, San Diego, CA, USA. 1990, pp. 91-99.

[9] P. Kumawat and N.Sharma, "Design and Development of Cost Measurement Mechanism for Re-Engineering Project Using Function Point Analysis," In R. Kamal, M. Henshaw and P. Nair (eds.), International Conference on Advanced Computing Networking and Informatics. Advances in Intelligent Systems and Computing, vol. 870. Springer, Singapore,2019.

[10] J. kisielnicki and A.M. Misiak, “Effectiveness of agile compared to waterfall implementation methods in it projects: analysis based on business intelligence projects”, Foundation of management, vol. 9, No. 1, pp. 273–286, 2017.

[11] A. Sahoo. D. Kung, and S. Gupta, “An Agile Methodology for Reengineering Object-Oriented Software,” Proceeding of 28th International Conference on Software Engineering & Knowledge Engineering: SEKE. California; USA, 2016.

[12] S. Chung, D.H. Won, S. H. Baeg and S. Park, “A Model-Driven Scrum Process for Service-Oriented Software Reengineering:mScrum4SOSR,” Proceeding of 2nd International Conference on Computer Science and its Applications, Korea (South), 2009, pp. 1-8.

[13] M.I. Cagnin, J. C. Maldonado and R.D. Penteado, “PARFAIT: Towards a framework-based agile reengineering process,” Proceedings of the Agile Development Conference (ADC): USA. 2003.pp. 22-31

[14] P.O. Adrian, “Aspect-Oriented Reengineering of an Object-oriented Library in a Short Iteration Agile Process,” Informatica, vol. 35, No.4, 2012.

[15] S.R. Chidamber and C.F. Kemerer, “A metrics suite for object-oriented design,” IEEE Transactions on Software Engineering, Vol. 20, No. 6, 1994, pp.476-493.

[16] M. North. Data mining for the masses. Global Text Project. August 2012.

[17] J. Singh, A. Gupta, and J. Singh, “Identification of requirements of software reengineering for JAVA projects,” Proceeding of International Conference on Computing, Communication, and Automation (ICCCA). Greater Noida; India, 2017, pp.931-934.

[18] J. Singh, K. Singh, and J. Singh, “Reengineering framework for open source software using decision tree approach,” International Journal of Electrical and Computer Engineering (IJECE), vol. 9, No. 3, 2019, pp.2041-2048.

[19] P. Serrador and J.K. Pinto,” Does Agile work? - A quantitative analysis of agile project success,” International Journal of Project Management, vol. 33,No. 5, 2015,pp.1040-1051.

[20] J. Kisielnicki and A.M. Misiak, “Effectiveness of agile compared to waterfall implementation methods in its projects analysis based on business intelligence projects,” Foundation of management, vol. 9 , No.1, 2017, pp. 273–286.

[21] scrumalliance.org. Scrum Alliances report-2017 [cited 2019 March 9] available from https://www.scrumalliance.org/learn-about-scrum/state-of-scrum/2018-state-of-scrum,.

[22] M. Cohan. Estimating and planning. Pearson Education.USA. 2006.

[23] V.R. Basili, L.C. Briand, and W.L. Melo, “Validation of object-oriented design metrics as quality indicators,” IEEE Transactions on Software Engineering, vol. 22, No.10, 1996, pp.751-761.

[24] J. Choudhary and U. Suman, “Story Points Based Effort Estimation Model for Software Maintenance,” Procedia Technology. Elsevier.vol.4, 2012, pp.761-765.