

Software Quality Estimation using Machine Learning: Case-based Reasoning Technique

Ekbal Rashid
Department of CS & E SOA
University,
Bhubaneswar, Orissa

Srikanta Patnaik
Department of CS & E SOA
University,
Bhubaneswar, Orissa

Vandana Bhattacharjee
Department of CS & E, Birla
Institute of Technology,
Ranchi, Jharkhand

ABSTRACT

Software quality estimation is one of the most interesting research areas in the domain of software engineering for last few decades. Large numbers of techniques and models have already been worked out in the area of error estimation. The aim of software quality estimation is to identify error prone tasks as the cost can be minimized with advance knowledge about the errors and this early treatment of error will enhance the software quality. In this paper we have explored a set of data in university setting. This paper advocates the use of case-based reasoning (i.e., CBR) to make a software quality estimation system by the help of human experts. CBR relies on historical information from similar past projects, whereby similarities are determined by comparing the projects, and key attributes. We have used different similarity measures to find the best method which increases estimation accuracy & reliability. This paper presents a work in which we have expanded our previous work [24]. The software is a console based application and thus does not use the GUI functions of the Operating System, which makes it very fast in execution. In order to obtain results we have used an indigenous tool for software quality estimation, run in c++ compiler.

General Terms

Software Engineering: Quality

Keywords

Software Quality estimation, CBR, Analogy, Similarity measure, Machine learning, Error.

1. INTRODUCTION

The degree to which software possesses attributes such as efficiency, usability, reliability, portability, reusability and maintainability is known as software quality [18]. Various software quality characteristics have been suggested by authors such as James McCall and Barry Boehm. There have been differences in opinion regarding the exact definitions of the qualities of good software. For example maintainability has been used to define the ease with which an error can be located and rectified in software. This definition also includes the ease with which changes can be incorporated in the software [19]. Estimation models in software engineering are used to predict some important attributes of future entities such as software development effort, software reliability, software quality, and productivity of programmers. Among such models, those estimating software effort have motivated considerable research in recent years [12]. Accurate and timely estimation of the development or maintenance cost of a software system is a critical activity in managing a software project. Due to the nature of the software engineering domain, it is important that

software effort estimation models should be able to deal with imprecision and uncertainty associated with such values. It is to serve this purpose that we propose our case-based estimation model for software quality estimation. We feel that case based models are particularly useful when it is difficult to define concrete rules about a problem domain in addition to this, expert advice may be used to supplement the existing stored knowledge. A case-based reasoning model was developed in [24] for software quality estimation.

The rest of the paper is organized as follows: section 2 gives a brief overview of the various related work, section 3 describes the overview of machine learning, section 4 describes the case-based reasoning approach and section 4.1 presents analysis. In section 5 we present similarity measures and section 6 presents Research Methodology, section 7 presents the calculation of errors, section 8 describes the program logic for CBR, section 9 presents Results & discussion and section 10 conclusions is presented. The snapshots (see snapshot 2 through snapshot 7) provides the details of data set which is used in this research paper.

2. BACKGROUND AND RELATED WORK

Many researchers have used soft computing approaches for software quality estimation. Zhong et. al in [22] have used unsupervised Learning techniques to build a software quality estimation system. Idri et al have implemented the COCOMO cost model using fuzzy logic in [1] and also a fuzzy logic based analogy estimation approach in [2-4]. T. M. Khoshgoftaar has described six software quality estimation techniques namely Regression Tree, Fuzzy Systems, Case-Based Reasoning, Rule-Based Systems, Multiple Linear Regression and Neural Networks [20]. Case-based reasoning has also been used by Kadoda et. al in [5]. They examine the impact of the choice of number of analogies when making estimations: They also look at different adaptation strategies. The analysis is based on a dataset of software projects collected by a Canadian software house. Their results show that choosing analogies is important but adaptation strategy appears to be less. For this reason they urge some degree of caution when comparing competing estimation systems and only modest numbers of cases. Myrtveit et al in [6] and Ganesan et. al in [7] have also studied case based approach to development effort estimation. Bhattacharjee et. al have proposed Expert Case Based Models in [10-17]. Rashid et. al emphasized on the importance of software quality estimation [24].

3. OVERVIEW OF MACHINE LEARNING

Machine learning deals with the issue of how to build programs that improve their performance at some task through experience [25]. Machine learning methods are used to predict or estimate software quality, software size, software development cost or

software effort [26]. Machine learning has been utilized in various problem domains.

Some typical applications of machine learning are [24]

- Optical character recognition
- Face detection
- Spam filtering
- Fraud detection
- Medical diagnosis
- Weather estimation

Major categories of machine learning techniques are [24]

- Case-based reasoning(CBR)
- Rule induction(RI)
- Neural networks(NN)
- Genetic algorithms(GA)
- Inductive logic programming(ILP)

4 .CASE-BASED REASONING

Case-Based reasoning is one of the most popular machine learning techniques. Case-based reasoning (CBR) is a problem solving paradigm that is fundamentally different from other major AI approaches, in that instead of relying solely on general knowledge of a problem domain it uses specific cases [23]. Instead of making association along generalized relationships between problem descriptors and conclusion, CBR utilizes the specific knowledge of previously experienced, concrete problem situation (cases). Finding a similar past case, and reusing it in the new problem situation: this is the technique of CBR to solve a new problem. A second important difference is that CBR is also an approach to incremental, sustained learning since a new experience is retained each time a problem has been solved, making it available for future problems.

Thus, the notion of case-based reasoning does not only denote a particular reasoning method, irrespective of how the cases are acquired, it also denotes a machine learning paradigm that enables sustained learning by updating the case base after a problem has been solved. Learning in CBR occurs as a natural by-product of problem solving. When a problem is successfully solved, the experience is retained in order to solve similar problems in the future. When an attempt to solve a problem fails, the reason for the failure is identified and remembered in order to avoid the same mistake in the future. Case-based reasoning prefers learning from experience, since it is usually easier to learn by retaining a concrete problem solving experience than to generalize from it.

Case-based estimation is one of the more attractive techniques in the software quality estimation field. Central tasks that all case-based reasoning methods have to deal with are to identify the current problem situation, find a past case similar to the new one, and use that case to suggest a solution to the current problem. Evaluate the proposed solution, and update the system by learning from this experience.

4.1. ANALYSIS

We can thus broadly categorize the four primary steps comprising a CBR estimation system as:

1. **RETRIEVE:** The most similar case or cases
2. **REUSE:** The information and knowledge in that case to solve the problem
3. **REVISE:** The proposed solution

4. **RETAIN :** The parts of this experience likely to be useful for future problem solving

A new problem is solved by retrieving one or more previously experienced cases, reusing the case in one way or another, revising the solution based on reusing a previous case, and retaining the new experience by incorporating it into the existing knowledge-base (case-base).

An initial description of a problem defines a new case. This new case is used to RETRIEVE a case from the collection of previous cases. The retrieved case is combined with the new case - through REUSE - into a solved case, i.e. a proposed solution to the initial problem.

Through the REVISE process this solution is tested for success, e.g. by being applied to the real world environment or evaluated by a teacher, and repaired if failed. During RETAIN, useful experience is retained for future reuse, and the case base is updated by a new learned case, or by modification of some existing cases.

As indicated in the figure 1, general knowledge usually plays a part in this cycle, by supporting the CBR processes. By general knowledge we here mean general domain-dependent knowledge, as opposed to specific knowledge embodied by cases. For example, in diagnosing a patient by retrieving and reusing the case of a previous patient, a model of anatomy together with causal relationships between pathological states may constitute the general knowledge used by a CBR system.

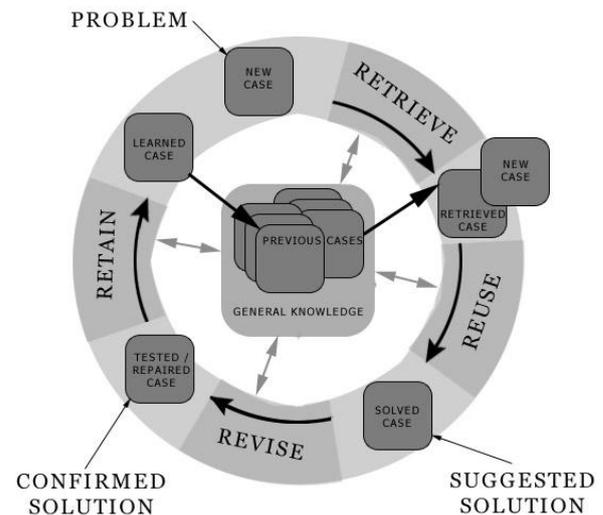


Figure 1: The CBR Cycle

5. SIMILARITY MEASURES

Suppose a record set P_1 of n fields has following values f_1, f_2, \dots, f_n for the n fields respectively. Similarly a record set P_2 of same type as P_1 with field values g_1, g_2, \dots, g_n . Then

The Euclidian distance (ED) of P_1 from P_2 is:

$$ED = \sqrt{(f_1 - g_1)^2 + (f_2 - g_2)^2 + \dots + (f_n - g_n)^2}$$

The Manhattan distance (MD) of P1 from P2 is:

$$MD = |abs(f_1 - g_1) + abs(f_2 - g_2) + \dots + abs(f_n - g_n)|$$

6. RESEARCH METHODOLOGY

The strategy chosen for the study were the students of computer science and engineering from the university campus. All students are provided with same level of guidance by instructors, support by the laboratory staffs, resources like computers, software etc.

In this study Data collected from students included the followings [24]:

- Number of lines of code(LOC)
- Number of functions(FUNC)
- Difficulty level(DL) of program (low , medium , high)
- Development Time
- Programmers experience(PEX)

The values for lines of code, number of functions, level of difficulty, experience and development time were collected from programs developed by students of the university over a period of 1 year. We have considered only post-graduate students in our study.

7. CALCULATION OF ERRORS

A knowledge base is created and maintained to store the cases against which the matching process has to be performed. Parameters related to the software are given as input and the error is predicted. The error estimation is done using varying similarity measures like Manhattan and Euclidean distance. The accuracy of estimates is evaluated by using the formula which is given below

Targeted parameter: TP

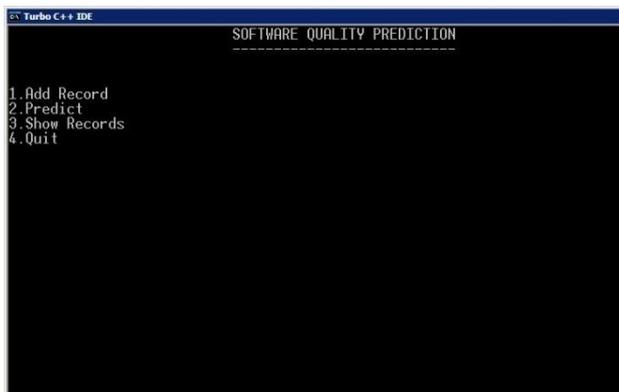
Actual parameter: AP

Error in estimation (EE) = (AP-TP)

Magnitude of Relative Error (MRE) = $abs(AP-TP) / AP$

8. PROGRAM LOGIC FOR CBR:

Step 1: When the file open is successful then user menu is shown (see snapshot 1).



Snapshot 1

Step 2: If file is opened successfully, records are shown one screen at a time after a small delay between them. The program also shows the position of the cases in the knowledge base and the total records that exists in the file (see snapshot 2 through snapshot 7)..

LOC	Func	Difficulty	LOC+Func.	Dev.Time	Exp.	Position
25	2	3	27	10	2	[POS->000]
54	2	2	56	20	2	[POS->001]
31	1	1	33	10	2	[POS->002]
17	2	1	19	10	2	[POS->003]
38	1	2	39	10	2	[POS->004]
27	2	3	29	15	2	[POS->005]
57	2	1	59	10	2	[POS->006]
28	2	1	30	5	2	[POS->007]
31	2	2	33	5	2	[POS->008]
46	1	2	47	15	2	[POS->009]
25	2	3	28	10	2	[POS->010]
55	2	2	57	15	2	[POS->011]
32	1	1	34	10	2	[POS->012]
16	2	2	18	5	2	[POS->013]
35	2	2	37	20	2	[POS->014]
23	2	3	25	10	2	[POS->015]
70	2	1	72	20	2	[POS->016]
35	3	1	38	15	2	[POS->017]
20	2	1	22	10	2	[POS->018]
35	1	2	36	15	2	[POS->019]

Snapshot 2

LOC	Func	Difficulty	LOC+Func.	Dev.Time	Exp.	Position
25	1	3	26	15	2	[POS->020]
30	1	1	31	15	2	[POS->021]
35	3	1	38	13	2	[POS->022]
20	2	1	22	10	2	[POS->023]
35	1	1	36	15	2	[POS->024]
27	2	3	28	12	2	[POS->025]
65	2	1	67	20	2	[POS->026]
35	3	1	38	15	2	[POS->027]
20	2	1	22	13	2	[POS->028]
46	2	1	48	12	2	[POS->029]
16	1	1	17	5	2	[POS->030]
32	2	1	34	5	2	[POS->031]
28	1	2	29	4	2	[POS->032]
25	1	3	26	4	2	[POS->033]
32	1	1	33	4	2	[POS->034]
40	1	2	41	10	2	[POS->035]
23	2	2	25	4	2	[POS->036]
18	2	2	20	10	2	[POS->037]
18	1	2	19	10	2	[POS->038]
32	1	2	33	15	2	[POS->039]

Snapshot 3

LOC	Func	Difficulty	LOC+Func.	Dev.Time	Exp.	Position
34	4	2	38	15	2	[POS->040]
15	1	2	17	13	2	[POS->041]
32	1	1	34	4	2	[POS->042]
28	1	1	29	4	2	[POS->043]
25	1	2	26	4	2	[POS->044]
32	1	1	33	4	2	[POS->045]
40	1	2	42	4	2	[POS->046]
16	1	1	17	4	2	[POS->047]
18	1	3	21	4	2	[POS->048]
17	1	1	18	5	2	[POS->049]
32	1	2	33	6	2	[POS->050]
26	1	1	27	6	2	[POS->051]
36	1	1	37	7	2	[POS->052]
19	1	1	20	7	2	[POS->053]
22	2	1	24	5	2	[POS->054]
23	1	2	24	7	2	[POS->055]
46	3	2	49	15	2	[POS->056]
22	4	1	26	8	2	[POS->057]
36	2	1	38	10	2	[POS->058]

Snapshot 4

LOC	Func	Difficulty	LOC+Func.	Dev.Time	Exp.	Position
30	1	2	31	15	2	[POS->060]
49	1	1	50	20	2	[POS->061]
35	2	1	37	13	2	[POS->062]
30	1	2	31	20	2	[POS->063]
23	2	2	25	20	2	[POS->064]
29	1	3	30	10	2	[POS->065]
40	1	3	43	15	2	[POS->066]
35	2	1	37	15	2	[POS->067]
31	1	1	32	20	2	[POS->068]
33	1	2	34	15	2	[POS->069]
22	1	1	24	10	2	[POS->070]
21	1	2	22	10	2	[POS->071]
21	2	1	23	10	2	[POS->072]
16	2	1	18	10	2	[POS->073]
38	1	3	41	15	2	[POS->074]
22	1	3	23	10	2	[POS->075]
47	3	1	50	15	2	[POS->076]
10	1	3	11	5	2	[POS->077]
40	1	1	42	12	2	[POS->078]
32	2	2	34	12	2	[POS->079]

Snapshot

LOC	Func	Difficulty	LOC*Func.	Dev.Time	Exp.	Position
45	2	1	47	10	2	[POS->080]
40	3	2	43	13	2	[POS->081]
40	2	2	42	13	3	[POS->082]
35	2	2	37	18	4	[POS->083]
45	2	2	47	11	4	[POS->084]
30	2	2	32	15	2	[POS->085]
25	2	2	27	10	3	[POS->086]
25	2	2	27	15	2	[POS->087]
30	2	2	32	14	3	[POS->088]
40	2	2	23	10	2	[POS->089]
20	1	2	42	14	1	[POS->090]
20	1	2	21	5	3	[POS->091]
30	1	2	31	14	1	[POS->092]
13	1	1	14	10	2	[POS->093]
15	1	2	15	5	2	[POS->094]
24	1	2	26	10	4	[POS->095]
25	2	1	27	10	2	[POS->096]
30	2	3	32	13	2	[POS->097]
60	1	2	63	15	4	[POS->098]
35	1	2	36	13	2	[POS->099]

Snapshot 6

LOC	Func	Difficulty	LOC*Func.	Dev.Time	Exp.	Position
34	3	2	37	10	2	[POS->100]
45	2	2	49	10	2	[POS->101]
30	2	2	33	20	2	[POS->102]
69	2	2	71	25	2	[POS->103]
60	2	2	62	15	2	[POS->104]
60	2	2	62	15	2	[POS->105]
66	2	2	68	24	2	[POS->106]
48	2	2	60	34	2	[POS->107]
34	2	2	36	10	2	[POS->108]
18	2	2	20	21	2	[POS->109]
27	2	2	39	30	2	[POS->110]
45	2	2	48	25	2	[POS->111]
70	2	2	72	23	2	[POS->112]

112 total records in file_

Snapshot 7

Step 3: This menu is used for manual data entry into the knowledge base using add record function (A)(see snapshot 8).

```

SOFTWARE QUALITY PREDICTION
-----
1. Add Record
2. Predict
3. Show Records
4. Quit
5.

Enter Lines Of Code : 44
Enter Number Of Functions : 50
Enter Difficulty Level : 2
Enter Development Time : 31
Enter Programmer's Experience : 2_
    
```

Snapshot 8

Step 4: pred () function is used for calculating magnitude of relative error. If the user selects the estimation option, the following options are displayed. (see snapshot 9).

```

SOFTWARE QUALITY PREDICTION
-----
1. Add Record
2. Predict
3. Show Records
4. Quit
5.

Select Prediction Method
1. Euclidean
2. Manhattan
    
```

Snapshot 9

9. RESULTS & DISCUSSION

We present the results obtained when applying the Case-based reasoning model to the data set. Both the Euclidean and Manhattan method was taken into consideration in terms of percentage of errors generated during execution of programs. It was noticed that the user is prompted if he is satisfied with the estimation or he wants to revise his input for further estimation. If the user is satisfied with the results, the program checks if the retrieved case is an exact match of the input case, if it is an exact match the program exits, or else it saves the case in the knowledge base for further use. If the error in estimation is less than 10% then the input record set is auto saved to the knowledge base. But if the error in estimation is more than 10% then the input record set must be revised then save to knowledge base for future solution. These data can be safely classified as high quality data.

In the Euclidean and Manhattan method the integer array that is passed is the address of the knowledge base in memory, user is prompted for various inputs such as Lines of code, number of functions, difficulty level, actual development time, programmers experience and their respective weightage. The distance values of each case is calculated from user's inputs the parameters of the software and exact/near matching case is retrieved from the knowledge base. The mean of such values is calculated and Magnitude of Relative error (MRE) is displayed on screen using pred () function . We also display the software quality relative to the LOC retrieved from Knowledge base (Q1) and LOC of the user(Q2). It can be seen in the output that Results are coming very good when applying the case-based reasoning model (see snapshot 10 through snapshot 11).

```

Enter Lines Of Code(1-100) & Weight(0.0-1.0) : 46 1
Enter Number Of Functions(0-45) & Weight(0.0-1.0): 3 1
Enter Difficulty Level(1/2/3) & Weight(0.0-1.0): 2 1
Enter Actual Development Time : 10
Enter Programmer's Experience & Weight: 3 1

+-----+-----+-----+-----+-----+-----+
| LOC | Functions | Difficulty | LOC*Functions | Exp. | | Predicted Dev. Time |
+-----+-----+-----+-----+-----+-----+
| 46  | 3         | 2         | 49             | 3    | | 10  <-- INPUT      |
| 46  | 3         | 2         | 49             | 3    | | 10  <-- RETRIEVED  |
+-----+-----+-----+-----+-----+-----+

MEAN=0.00%          Error Prediction=0.008929
Record No. : 101
Quality
Q1(Predicted from KBS)=0.000000          Q2(Input from User)=0.000000
Do you want to continue ?(y/n)n
    
```

Snapshot 10

```

Enter Lines Of Code(1-100) & Weight(0.0-1.0) : 74 1
Enter Number Of Functions(0-45) & Weight(0.0-1.0): 3 1
Enter Difficulty Level(1/2/3) & Weight(0.0-1.0): 2 1
Enter Actual Development Time : 16
Enter Programmer's Experience & Weight: 3 1

+-----+-----+-----+-----+-----+-----+
| LOC | Functions | Difficulty | LOC*Functions | Exp. | | Predicted Dev. Time |
+-----+-----+-----+-----+-----+-----+
| 74  | 3         | 2         | 77             | 3    | | 16  <-- INPUT      |
| 70  | 2         | 1         | 72             | 3    | | 20  <-- RETRIEVED  |
+-----+-----+-----+-----+-----+-----+

MEAN=25.00%          Error Prediction=0.008929
Record No. : 16
Quality
Q1(Predicted from KBS)=0.100000          Q2(Input from User)=0.094595
Do you want to continue ?(y/n)n
Record Added!
    
```

Snapshot 11

If the user wants to revise the input the user menu is shown once again (see snapshot 12).



Snapshot 12

10. CONCLUSION

In this paper we have used two similarity measures namely Euclidean distance and Manhattan distance. For their efficacy in determining errors, the low error programs detected by these methods may help to design high quality software (error free programs). In this research we have shown the strong motivation behind the use of CBR as a tool for solving the problem which has been given as snapshot in this paper. As part of our ongoing work, increasing the volume of database is another objective. The larger the database more likely the results are to be accurate. We are collecting data from different categories of students and viewing different parameters. In this research, students programs were the basic target of study.

REFERENCES

- [1] A. Idri, L.Kjiri, and A Abran. (2000), "COCOMO Cost Model Using Fuzzy Logic", In *Proceedings of the 7th International Conference on Fuzzytheory and Technology*, pp.219-223. Atlantic City, NJ, USA.
- [2] A. Idri and A Abran. (2000b), "Towards A Fuzzy Logic Based Measures for Software Project Similarity", In *Proceedings of the 6th Maghrebian Conference on Computer Sciences*, pp. 9-18, Fes Morocco.
- [3] A. Idri and A. Abran. (2001), "A Fuzzy Logic Based Measures For Software Project similarity: Validation and Possible Improvements", In *Proceedings of the 7th International Symposium on Software Metrics*, pp. 85-96, England, UK, IEEE.
- [4] A. Idri , A. Abran and T.M. Khoshgoftaar .(2001c), " Fuzzy Analogy: Anew Approach for Software Cost Estimation", In *Proceedings of the 11th International workshop on software Measurements*, pp.93-101, Montreal, Canada.
- [5] G.Kadoda, M Cartwright, L Chen, and M.shepperd.(2000), "Experiences Using Case- Based Reasoning to Predict Software Project Effort", In *Proceeding of EASE*, p.23-28, Keele,UK.
- [6] I. Myrtveit and E. Stensrud. (1999), "A Controlled Experiment to Assess the Benefits of Estimating with Analogy and Regression Models", *IEEE transactions on software Engineering*, vol 25,no. 4, pp. 510-525.
- [7] K. Ganeasn, T.M. Khoshgoftaar, and E. Allen. (2002), "Case-based Software Quality Estimation", *International journal of Software Engineering and Knowledge Engineering*, 10 (2), pp. 139-152.
- [8] L. Angelis and I Stamelos. (2000), "A Simulation Tool for Efficient Analogy Based Cost Estimation", *Empirical software Engineering*, vol. 5, no. 1, pp.25-68.
- [9] M. Shepperd C. Schofield, and B. Kitchenham. (1996), "Effort Estimation using Analogy", In *Proceeding of the 18th International Conference on Software Engineering*, pp.170-178, Berlin.
- [10] S. Kumar and V.Bhattacharjee,(2005),"Fuzz logic based Model for Software cost Estimation ",In *Proceedings of the international Conference on information Technology, Nov'05*, PCTE, Ludhiana India.
- [11] S. Kumar and V.Bhattacharjee,(2007),"Analogy and Expert Judgment: A Hybrid Approach to Software Cost Estimation", In *Proceedings of the National Conference on information Technology: Present practice and Challenge, Sep'0* , New-Delhi, India.
- [12] V. Bhattacharjee and S. Kumar,(2004),"Software cost estimation and its relevance in the Indian software Industry", In *Proceedings of the International Conference on Emerging Technologies IT Industry, Nov'05*, PCTE, Ludhiana India..
- [13] V. Bhattacharjee and S .Kumar,(2006),"An Expert- Case Based Frame work for Software Cost Estimation", In *Proceedings of the National Conference on Soft Computing Techniques for Engineering Application (SCT-2006)*, NIT Rourkela.
- [14] V. Bhattacharjee,(2006),"The Soft Computing Approach to Program Development Time Estimation In *Proceeding of the International Conference on Information Technology, ICIT 06, Dec'06, Bhubneshwar, India, IEEE Computer Society*.
- [15] V. Bhattacharjee, S. Kumar and E. Rashid ,A Case Study on Estimation of Software Development Effort" In *Proceedings on International Conference on Advanced Computing Technologies(ICAICT-2008)*, Gokaraju Rangaraju Institute of Engg & Technology, Hyderabad, India,p.no.161-164.
- [16] V. Bhattacharjee, S. Kumar and E. Rashid ,"Case Based Estimation Model using Project Feature Weights" In *Proceedings of The National Seminar on Recent Advances on Information Technology(RAIT-2009)*,Department of Computer Science and Engg, ISMU, Dhanbad., p.no246-252
- [17] E. Rashid, V. Bhattacharjee, S. Patnaik, "The Application of Case-Based Reasoning to Estimation of Software Development Effort". *International Journal of Computer Science and Informatics (IJCSI) ISSN 2231 –5292, Vol 1 Issue 3* pp 29-34 Feb 2012.
- [18] Deepak Gupta,Vinay Kr Goyal,Harish Mittal,"Comparative study of soft computing techniques for software quality model",*International Journal of software Engineering Research&Practices Vol.1.Issue 1,Jan,2011*.
- [19] Bob Hughes & Mike Cotterell "software Project Management", Tata McGraw-Hill.
- [20] Khoshgoftaar, T. M., Cukic, B. and Seliya, N. "Comparative Study of the Impact of Underlying Models on Module-Order Model Performances", 8th IEEE International Symposium on Software Metrics, Boca Raton, Florida, USA, 161, 2002.

- [21] E. Rashid, S. Patnaik, V. Bhattacharjee, “Strategies Towards Improving Software Code Quality in Computing”, *International journal of Engineering Research and Applications(IJERA)* ISSN 2248-9622, Vol 2 Issue 3 pp 2253-2258 with Impact factor 0.68
- [22] Shi Zhong, Taghi M. Khoshgoftaar and Naeem Selvia “Unsupervised Learning for Expert-Based Software Quality Estimation”. *Proceeding of the Eighth IEEE International Symposium on High Assurance Systems Engineering (HASE’04)*
- [23] Aamodt, A. & E. Plaza. 1994. In: *AI Communications*. IOS Press, Vol. 7:1, pp39-59
- [24] Ekbal Rashid, Srikanta Patnaik, Vandana Bhattacharjee “A Survey in the Area of Machine Learning and Its Application for Software Quality Estimation” has been published in *ACM SigSoft* ISSN 0163-5948, volume 37, number 5, September 2012, <http://doi.acm.org/10.1145/2347696.2347709> New York, NY, USA.
- [25] T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.
- [26] Du Zhang and Jeffrey J.P. Tsai. “Machine Learning and Software Engineering”. *Proceeding of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI’02)*.