Intelligent Water Drops (IWD) Algorithm for COQUAMO Optimization

Abdulelah G. Saif, Safia Abbas, and Zaki Fayed, *Member, IAENG*

Abstract*—Software quality estimation is one of essential aspects in software projects. Accurate quality estimates are necessary for goodly developing software systems. Many estimation methods have been proposed. Among those methods, COQUAMO, the model used to estimate the quality of the software project in defects/KSLOC (or some other unit of size). Nowadays, estimation models are based on neural network, the fuzzy logic modeling etc. for accurately estimate software development effort, time and quality. As, neural networks design have not clear guidelines and fuzzy logic approach usage is more difficult, a meta-heuristic Intelligent Water Drops (IWD) algorithm can offer some improvements in accuracy for software quality estimation. This work adapts the IWD algorithm for optimizing the current coefficients of COQUAMO model to achieve more accurate estimation of software development quality. The experiment has been conducted on NASA 93 software projects. This work is the first one used to optimize COQUAMO.

Index Terms— **COQUAMO, IWD algorithm, Software quality estimation**

I. INTRODUCTION

A software project that is completed on time, within budget, and delivers a quality product that satisfies users and meets requirements is said to be successful. However, many software projects fail. Only a third of all software development projects were successful, in terms of they met budget, schedule, and quality goals as a report given by the Standish Group states [1]. Most project fails usually are due to the planning and estimation steps, not due to the implementation steps. Several studies have been done during the last decade, for finding the reason of the software projects failure. 2100 internet sites were searched extensively by Galorath et al. who found 5000 reasons for the software project failures. Among the found reasons, insufficient requirements engineering, poorly planned project, suddenly decisions making at the early stages of the project and inaccurate estimations were the most important reasons [2]. Therefore, accurate software cost, time and quality estimation is necessary and is critical to both developers and customers. Software cost estimation focuses on the time and the effort required to complete a software project. Software cost estimation starts at the proposal state and continues throughout the life time of a project [3]. The human-effort occupies the large part of software development cost and most cost estimation methods focus on this aspect and give estimates in terms of person-month [4]. Some software defects are unavoidable during software development, even if accurate planning, well documentation and proper process control are performed carefully. These software defects affect the quality of software product which might be the main cause of project failure [9]. Therefore, in order to manage budget, schedule and quality of software projects, several software estimation methods have been developed. Among those methods, COCOMO II is the most widely used model for estimating the effort in person-month and the time in months for the whole software project and also at different stages, and COQUAMO is the model used to estimate the quality of the software project in terms of defects/KSLOC (or some other unit of size). Nowadays, most estimation models are based on neural network, genetic algorithm, the fuzzy logic modeling etc. for accurately estimate software development effort, time and quality. As, neural networks have not clear guidelines for design and fuzzy logic approach usage is more difficult, the meta-heuristic intelligent water drops (IWD) algorithm can offer some improvements in accuracy for software quality estimation. This work adapts the IWD algorithm for optimizing the current coefficients of COQUAMO model to achieve more accurate estimation of software development quality. The experiment has been conducted on NASA 93 software projects. This work is the first one used to optimize COQUAMO.

The rest of the paper is organized as follow: section II related works, section III COQUAMO model, section IV dataset description, section V IWD algorithm, section VI results analysis and section VII discusses and concludes the paper.

II. RELATED WORK

There are many prediction models that can be used to predict software defects such as machine learning based models (artificial neural networks (ANN), Bayesian belief networks (BBN), reinforcement learning (RL), genetic algorithms (GA), genetic programming (GP) and decision trees) [9] and fuzzy logic models [2] etc. However each one has its own advantages and disadvantages and each one can be used for specific projects at different stages [9]. Because COCOMO II is the most widely used and standard model for estimating the effort in person-month and the time in months for a software project at different stages [4] and COQUAMO [10][11] is an extension of it, COQUAMO

Manuscript submitted July 10, 2015; revised July 22, 2015. The authors gratefully acknowledge the support of Ain Shams University and Yemen government in supporting them.
Abdulelah Ghaleb Farhan Saif is Ph.D student at Ain Shams University, Egypt (phone: 00201154415035; abdulelah.saif1980@gmail.com).
Safia Abbas Mahmoed Abbas is lecturer at Ain Shams University, Egypt (Safia_abbas@yahoo.com).
Zaki Taha Ahmed Fayed is Emeritus Professor at Ain Shams University, Egypt (ZFayed@hotmail.com).

ISBN: 978-988-19253-6-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

model will deserve much attention to improve it.

Because today's project quality evaluation based on old coefficients of COQUAMO model may not match the required accuracy, therefore by calibration, the accuracy of results in this method will be increased and the aim of this research is to use IWD algorithm to optimize the current COQUAMO model coefficients to achieve accurate software quality estimation and reduce the uncertainty of COQUAMO coefficients using IWD algorithm.

III. COQUAMO MODEL

Constructive quality model (COQUAMO), which is shown in figure 1, is an extension of the existing constructive cost model (COCOMO II) and consists of two sub-models; defects introduction (DI) sub-model and defects removal (DR) sub-model.

A. Defect Introduction (DI) Sub-Model

The DI sub-model's inputs include source lines of code and/or function points FPs as the sizing parameter, adjusted for both reuse and breakage, and a set of 21 multiplicative DI-drivers divided into four categories, platform, product, personnel and project. These 21 DI-drivers are a subset of the 22 cost parameters required as input for COCOMO II. Development flexibility FLEX driver has no effect on defect introduction and thus here its values for rating are set to 1.

The decision to use these drivers was taken after the author did an extensive literature search and did some behavioral analyses on factors affecting defect introduction. The outputs of DI sub-model are predicted number of non-trivial defects of requirements, design and code introduced during development life cycle; where non-trivial defects include:

- Critical (causes a system crash or causes a serious damage or jeopardizes personnel)
- High (causes impairment of critical system functions and no workaround solution exists)
- Medium (causes impairment of critical system function, though a workaround solution does exist).

Based on expert-judgment, an initial set of values to each of ratings of the DI-drivers that have an effect on the number of defects introduced and overall software quality were proposed and we are used them in our implementation.

B. Defect Removal (DR) Sub- Model

The aim of the defect removal (DR) model is to estimate the number of defects removed by several defect removal activities, namely automated analysis AUTA, people reviews PEER and execution testing and tools EXTT. The DR model is a post-processor to the DI model. Each of these three defect removal profiles removes a fraction of the requirements, design and coding defects introduced from DI model. Each profile has 6 levels of increasing defect removal capability, namely 'Very Low', 'Low', 'Nominal', 'High', 'Very High' and 'Extra High' with 'Very Low' being the least effective and 'Extra High' being the most effective in defect removal.

To determine the defect removal fractions (DRF) associated with each of the six levels (i.e. very low, low, nominal, high, very high, extra high) of the three profiles (i.e. automated analysis, people reviews, execution testing and tools) for each of the three types of defect artifacts (i.e. requirements defects, design defects and code defects), the author conducted a 2-round Delphi and we used the values of DRF resulted from 2-round Delphi in our implementation.

The inputs of DR sub-model include software size in thousand source lines of code KSLOC and/or function points, defect removal profiles levels and number of non-trivial defects of requirements (Req), design (Des) and code (Code) from DI model. For more details about COQUAMO, see [10] [11].

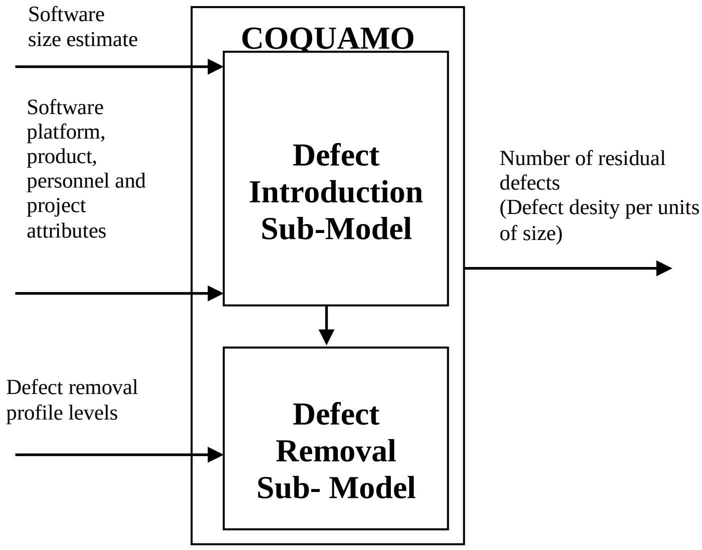

Fig. 1. Constructive quality model (COQUAMO).

For each of the three artifacts:
Estimated introduced defects in requirements:

$$DI^{Req} = A1.\ (Size)^{B1}.\ \prod_{j=1}^{21} (DI - Driver)\, j\, \mathrm{Re}\, q) \qquad (1)$$

Estimated residual defects in Requirements:

$$DR^{Req} = C1.\ DI^{Req}.\ \prod_{r=1}^{3} (1 - DRF)\, r\, \mathrm{Re}\, q \qquad (2)$$

Estimated introduced defects in design:

$$DI^{Des} = A2.\ (Size)^{B2}.\ \prod_{j=1}^{21} (DI - Driver)\, jDes) \qquad (3)$$

Estimated residual defects in design:

$$DR^{Des} = C2.\ DI^{Des}.\ \prod_{r=1}^{3} (1 - DRF)\, rDes \qquad (4)$$

Estimated introduced defects in code:

$$DI^{Code} = A3.\ (Size)^{B3}.\ \prod_{j=1}^{21} (DI - Driver)\, jCode) \qquad (5)$$

Estimated residual defects in code:

$$DR^{Code} = C3.\ DI^{Code}.\ \prod_{r=1}^{3} (1 - DRF)\, rCode \qquad (6)$$

A1, *A2*, *A3*, *C1*, *C2* and *C3* are the multiplicative calibration constants for each artifact. *Size* is the size of the software project measured in terms of KSLOC (thousands of source lines of code, function points FPs or any other unit of size), here KSLOC is converted to FPs as software size

ISBN: 978-988-19253-6-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

measure by assuming c language is used for implementation. *B1*, *B2* and *B3* account for economies / diseconomies of scale. *(DI-driver)jReq* , *(DI-driver)jDes* and *(DI-driver)jCode* are the defect introduction driver for each artifact and the j^{th} factor.

r = 1 to 3 for each DR profile, namely automated analysis, people reviews, execution testing and tools.

DRF_{rReq}, DRF_{rDes} and DRF_{rCode} are Defect Removal Fraction for defect removal profile r and artifact type (Req, Des and Code).

IV. DATASET DESCRIPTION

Experiments have been conducted on NASA 93 data set found in [5]. The dataset consist of 93 completed projects with its size in kilo line of code (KLOC) and actual quality in defects/ KLOC. Here KSLOC is converted to FPs as software size measure by assuming c language is used for implementation. Multipliers (DI-Drivers) and scale factors rating from Very Low to Extra High are also given in the dataset. Defects removal activities levels (ratings) are not found in NASA 93 dataset, so the ratings are assumed to be of 'Nominal' rating.

In this data set, there is no classification of defects into requirements (Req), design (Des) and Code (Code) defects, So the total defects of each project in the dataset are converted into Req, Des and Code defects according to Jones report [12] such that documentation defects =0.60 per function point FP, requirements defects=1 per FP, design defects =1.25 per FP , code defects=1.75 per FP and bad fixes defects=0.40.

Therefore, Req, Des and Code defects for each project j (Pr_j) in the data set are calculated as follow:

$$Pr_j \text{ Req defects} = \frac{\text{total defects of } Pr_j}{128} *(1+0.20) \quad (7)$$

$$Pr_j \text{ Des defects} = \frac{\text{total defects of } Pr_j}{128} *(1.5+0.20) \quad (8)$$

$$Pr_j \text{ Code defects} = \frac{\text{total defects of } Pr_j}{128} *(1.75+0.20) \quad (9)$$

Where, documentation defects are divided equally among artifacts by assumption (0.20 for each). 128 is a factor of c language to convert SLOC into FPs. Bad fixes defects are related to DI-drivers.

V. INTELLIGENT WATER DROPS (IWD) ALGORITHM

In nature, flowing water drops are mostly seen in rivers, which form huge moving swarms. The paths that a natural river follows have been created by a swarm of water drops.

while a natural water drop flows from one point of a river to the next point in the front, Three changes occur during this transition: the water drop velocity is increased, the water drop soil is increased, and in between these two points, soil of the river's bed is decreased.

Based on these observations, the Intelligent Water Drops have been introduced. These Intelligent Water Drops or IWDs flow in a graph of a given optimization problem. Then, the resulting effect is that the best solution is obtained for the problem [6, 7].

A. *Assumption and Representation*

IWD is basically developed for *combinatorial optimization problems*, with only one paper [8] modifies it to be used for *continuous optimization problems* by using the binary coding of edges. To use IWD for *continuous optimization problems* such as optimizing the COQUALMO model and maintaining the original structure of IWD, the coefficients of COQUALMO model, *A1*, *B1*, *C1*, *A2*, *B2*, *C2*, *A3*, *B3* and *C3*, are assumed to be represented by the graph by adding virtual nodes numbered from 0 to 9 and connecting those nodes to each coefficient as in figure 2, where i= 1..3.

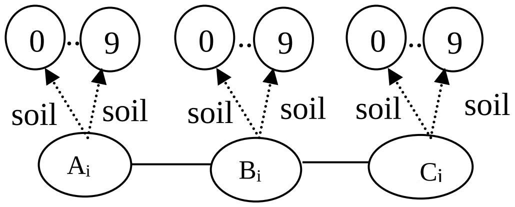

Fig. 2. Coefficients representation.

Each of above nine coefficients is expressed by 4 digits which are chosen among 10 digits by IWD algorithm according to minimum probabilities. First digit is integral part of a coefficient and the remaining 3 are fractions part.

The soils are placed on the edges between coefficients and digits as in the figure 2.

B. *The Proposed IWD Algorithm*

To optimize the COQUALMO model coefficients, The main steps of proposed IWD algorithm are in figure 3.

VI. RESULT ANALYSIS

IWD parameters initial values: *Number of IWDs=5, Initial Soil=10000 , velocity=100, local and global soil updating parameters=0.9,* $a_v = 1, b_v = 0.01, c_v = 1, a_s = 1, b_s = 0.01$ and $c_s = 1$.

The best result is achieved using 10000 iterations, and a solution set is received from which the best solution is chosen i.e. a solution with the best fitness function value (*FitnessAll*). The final best solution obtained for coefficients are: 0405|1028|0717|0405|1028|0717|0724|105|0782|. According to this solution, the resulting optimized COQUALMO model coefficients are the following: *A1= 0.405, B1= 1.028, C1= 0.71, A2= 0.405, B2= 1.028, C2= 0.717, A3= 0.724, B3= 1.05and C3= 0.782.* Current COQUALMO model coefficients are all equal 1.

Tables, table I and table II, show the comparison among actual and estimated Req, Des and Code defects/FP obtained from COQUALMO model using its current coefficients and using optimized coefficients by IWD, respectively for the first ten project dataset with their estimated project size.

Table III shows the comparison among the actual and estimated Req, Des and Code defects/FP for the first ten project dataset using optimized and current COQUAMO model coefficients with their estimated project size.

The graphical comparison described in table I and table II is shown in figure 4 and figure 5, respectively.

The graphical comparison described in table III is shown in figure 6, figure 7 and figure 8.

ISBN: 978-988-19253-6-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

1. Set parameters and determine dataset
2. Initialize the soils of virtual edges between coefficients and their digits.
3. While (termination condition not met) do
4. IWDs are placed on the first node A1 and move to the next until node C3 is reached.
5. IWDs choose 4 digits among 10 digits as values for all coefficients according to minimum probabilities and add the digits to their visited lists. If there is no improvement in one of the fitness functions (*FitnessCode*) in step 10, IWDs choose 4 digits for each coefficient randomly.
6. IWDs update their velocity
7. IWDs update soils on edges between coefficients and chosen digits and load some soils according to IWD algorithm equation 4 in [6].
8. Each IWD i calculate estimated Req, Des and Code defects for each project j in the dataset using the values of coefficients chosen by them.
9. Each IWD i calculates *Magnitude of Relative Error (MRE)* for each project j, the equation used for Req, Des and Code, respectively are:

$$FitnessReqij = |\, ActualReqj - EstimatedReqij \,| \,/\, ActualReqj \qquad (10)$$

$$FitnessDesij = |\, ActualDesj - EstimatedDesij \,| \,/\, ActualDesj \qquad (11)$$

$$FitnessCodeij = |ActualCodej - EstimatedCodesij \,| \,/\, ActualCodej \qquad (12)$$

10. The fitness functions *(Mean Magnitude of Relative Error MMRE)* for each artifact are calculated as the average value of all projects specific fitness values calculated during steps 8 and 9 which depends on the difference between real and estimated Req, Des and Code defects. So, the fitness functions values should be minimized.

$$FitnessReq = 1/n * \sum_{j=1}^{n} FitnessReqij \qquad (13)$$

$$FitnessDes = 1/n * \sum_{j=1}^{n} FitnessDesij \qquad (14)$$

$$FitnessCode = 1/n * \sum_{j=1}^{n} FitnessCodeij \qquad (15)$$

$$FitnessAll = (FitnessReq + FitnessDes + FitnessCode)/3 \qquad (16)$$

11. Find the iteration best solution (optional).
12. Update the soils of virtual edges that form current best solution according to IWD equation 6 in [6] (optional).
13. end while.
14. Return the values of coefficients and the estimated Req, Des and Code defects.

i - the IWD number, j – the project number, *ActualReqj* - is actual software Req defects, *EstimatedReqij* - is the estimated software Req defects, using IWD i, *ActualDesj* - is actual software Des defects, *EstimatedDesij* - is the estimated software Des defects, using IWD i, *ActualCodej* - is actual software Code defects and *EstimatedCodesij* - is the estimated software Code defects, using IWD i.

Fig. 3. Proposed IWD algorithm.

TABLE I: ESTIMATED DEVELOPMENT DEFECTS VALUES BY CURRENT COQUAMO COEFFICIENTS

Pr. No	FPs	Actual Req Defects/FP	Req Defects /FP Calculated by Current COQUAMO coefficients	Actual Des Defects/FP	Des Defects/FP Calculated by Current COQUAMO coefficients	Actual Code Defects/FP	Code Defects /FP Calculated by Current COQUAMO coefficients
1	202.34375	7.575	27.18968	9.153125	29.48876	12.309375	17.5053
2	192.1875	7.190625	25.82495	8.6886719	28.00863	11.684766	16.62665
3	60.15625	2.25	8.083419	2.71875	8.766929	3.65625	5.204278
4	64.0625	2.4	8.608317	2.9	9.33621	3.9	5.542218
5	75.78125	2.83125	10.18301	3.4210938	11.04405	4.6007813	6.556038
6	17.1875	0.646875	2.309548	0.7816406	2.504837	1.0511719	1.486937
7	27.34375	1.021875	3.674282	1.2347656	3.984968	1.6605469	2.365581
8	520.3125	19.471875	69.91633	23.528516	75.82825	31.641797	45.01362
9	58.59375	2.11875	6.333148	2.5601563	8.02877	3.4429688	4.437561
10	156.25	5.30625	17.9552	6.4117188	18.78181	8.6226563	11.03867

TABLE II: ESTIMATED DEVELOPMENT DEFECTS VALUES BY OPTIMIZED IWD COEFFICIENTS

Pr_No	FPs	Actual Req Defects/FP	Req Defects/FP Calculated by Optimized IWD coefficients	Actual Des Defects/FP	Des Defects/FP Calculated by Optimized IWD coefficients	Actual Code Defects/FP	Code Defects /FP Calculated by Optimized IWD coefficients
1	202.34375	7.575	9.161126	9.153125	9.935763	12.309375	12.92467
2	192.1875	7.190625	8.688764	8.6886719	9.42346	11.684766	12.24437
3	60.15625	2.25	2.632626	2.71875	2.855233	3.65625	3.616345
4	64.0625	2.4	2.808519	2.9	3.045999	3.9	3.863306
5	75.78125	2.83125	3.337936	3.4210938	3.620182	4.6007813	4.608556
6	17.1875	0.646875	0.726252	0.7816406	0.787661	1.0511719	0.970506
7	27.34375	1.021875	1.170519	1.2347656	1.269495	1.6605469	1.580251
8	520.3125	19.471875	24.18846	23.528516	26.23376	31.641797	34.84195
9	58.59375	2.11875	2.061074	2.5601563	2.612901	3.4429688	3.079514
10	156.25	5.30625	6.006085	6.4117188	6.282588	8.6226563	8.045497

ISBN: 978-988-19253-6-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE III: COMPARISON AMONG REQ, DES AND CODE DEFECTS VALUES VS. SIZE BY IWD AND COQUAMO

Pr_No	FPs	Actual Req Defects/FP	Req Defects /FP Calculated by Optimized IWD coefficients	Req Defects/FP Calculated by Current COQUAMO coefficients	Actual Des Defects/FP	Des Defects/FP Calculated by Optimized IWD coefficients	Des Defects/FP Calculated by Current COQUAMO coefficients	Actual Code Defects/FP	Code Defects/FP Calculated by Optimized IWD coefficients	Code Defects /FP Calculated by Current COQUAMO coefficients
1	202.34375	7.575	9.161126	27.18968	9.153125	9.935763	29.48876	12.309375	12.92467	17.5053
2	192.1875	7.190625	8.688764	25.82495	8.6886719	9.42346	28.00863	11.684766	12.24437	16.62665
3	60.15625	2.25	2.632626	8.083419	2.71875	2.855233	8.766929	3.65625	3.616345	5.204278
4	64.0625	2.4	2.808519	8.608317	2.9	3.045999	9.33621	3.9	3.863306	5.542218
5	75.78125	2.83125	3.337936	10.18301	3.4210938	3.620182	11.04405	4.6007813	4.608556	6.556038
6	17.1875	0.646875	0.726252	2.309548	0.7816406	0.787661	2.504837	1.0511719	0.970506	1.486937
7	27.34375	1.021875	1.170519	3.674282	1.2347656	1.269495	3.984968	1.6605469	1.580251	2.365581
8	520.3125	19.471875	24.18846	69.91633	23.528516	26.23376	75.82825	31.641797	34.84195	45.01362
9	58.59375	2.11875	2.061074	6.333148	2.5601563	2.612901	8.02877	3.4429688	3.079514	4.437561
10	156.25	5.30625	6.006085	17.9552	6.4117188	6.282588	18.78181	8.6226563	8.045497	11.03867

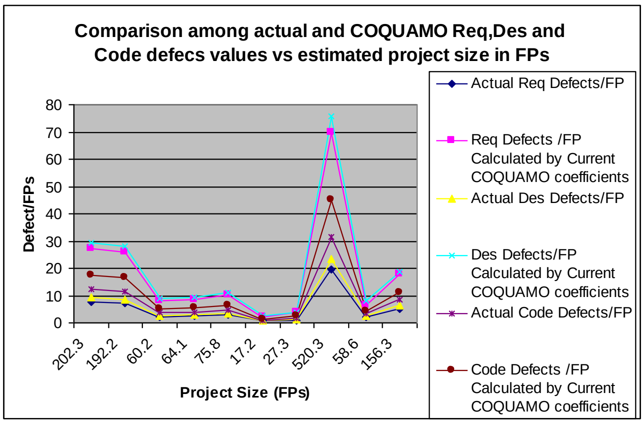

Fig. 4. Comparison among defects values by COQUAMO.

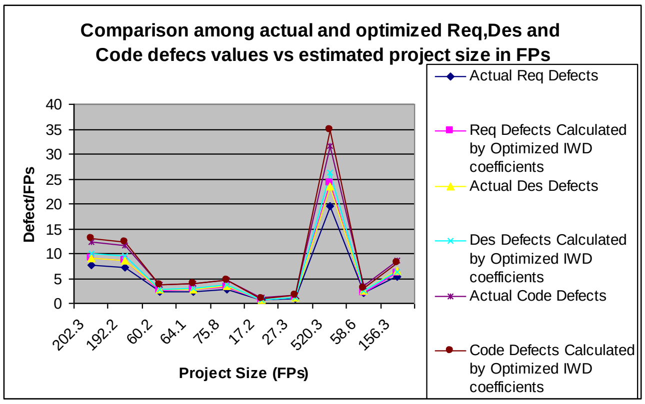

Fig. 5. Comparison among defects values by IWD.

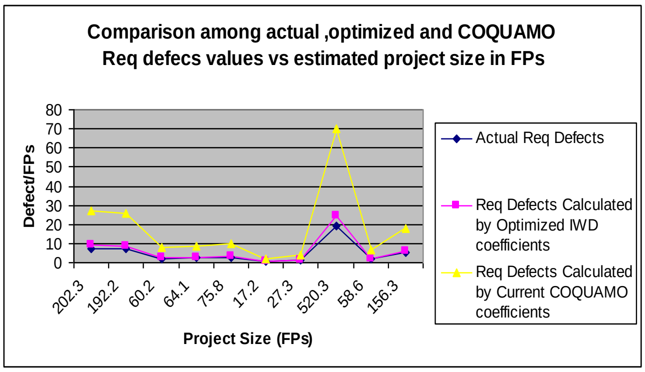

Fig. 6. Comparison among Req defects values by IWD and COQUAMO.

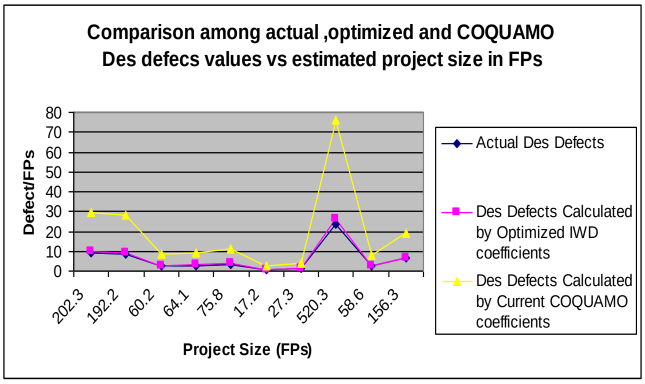

Fig. 7. Comparison among Des defects values by IWD and COQUAMO.

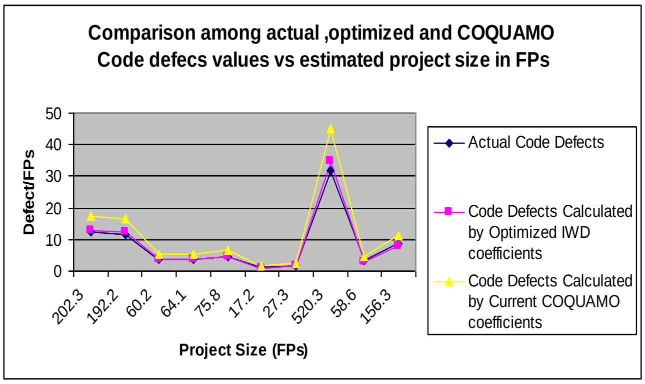

Fig. 8. Comparison among Code defects values by IWD and COQUAMO.

Table IV compares the *MMRE (Mean Magnitude of Relative Error)* and *PRED (.25)* (prediction (0.25) which shows the performance of IWD and COQUAMO in estimating the Req, Des and Code defects for the whole dataset.

ISBN: 978-988-19253-6-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

$$MMRE = \frac{1}{n} * \sum_{j=1}^{n} \frac{| \text{Actual}_j - \text{Estimated}_j |}{\text{Actual}_j} \quad (17)$$

$$MREj = \frac{| \text{Actualj} - \text{Estimatedj} |}{\text{Actualj}} \quad (18)$$

$$PRED(p) = k / n \quad (19)$$

k is the number of projects where MRE is less than or equal to p, and n is the total number of projects.

The graphical comparison described in table IV is shown in figure 9.

TABLE IV: PERFORMANCE MEASURE COMPARISON

Results	**IWD**	**COQUAMO**
MMRE_Req	0.129249	1.92676
MMRE_Des	0.062069	1.951845
MMRE_Code	0.059168	0.32434
Total MMRE	0.083495	1.400982
PRED_Req (0.25)	0.913978	0
PRED_Des (0.25)	0.989247	0
PRED_Code (0.25)	0.978495	0.204301
Total PRED (0.25)	0.960573	0.0681

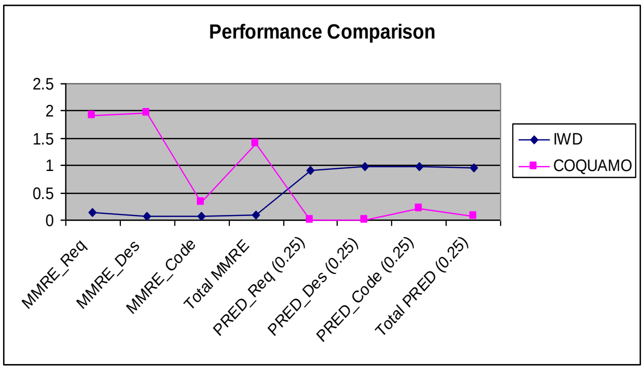

Fig. 9. Performance measure comparison.

From table IV and figure 9, the *MMRE* of IWD for Req, Des and Code defects is lower than that of COQUAMO and *PRED(0.25)* of IWD for Req, Des and Code defects is larger than that of COQUAMO.

It shows clearly that optimized coefficients by IWD algorithm produces more accurate results than the old coefficients. So, IWD algorithm can offer some significant improvements in accuracy and has the potential to be a valid additional tool for the software quality estimation

VII. DISCUSSION and CONCLUSION

This paper adapts IWD algorithm for optimizing COQUAMO coefficients. The proposed algorithm is tested on NASA 93 dataset and the obtained results are compared with the ones obtained using the current COQUAMO model coefficients. The proposed model is able to provide good estimation capabilities. It is concluded that

- By having the appropriate statistical data describing the software development projects, IWD based coefficients can be used to produces better results in comparison with the results obtained using the current COQUAMO model coefficients.
- The results show that, in the sample projects taken from the dataset, the results obtained using the coefficients optimized with the proposed algorithm are better than the ones obtained using the current coefficients.
- The results also show that in the sample projects taken from the dataset, the results obtained using the coefficients optimized with the proposed algorithm are close to the real defects values.
- The results also show that in the whole dataset, the *MMRE* of IWD is less than that of COQUAMO and *PRED(0.25)* is larger than that of COQUAMO.

In the future work or the next paper, we adapt PDBO algorithm for optimizing the coefficients of COQUAMO and compare it with IWD algorithm.

REFERENCES

[1] Gary B. Shelly, Harry J. Rosenblatt, "Systems Analysis and Design Ninth Edition", Shelly Cashman Series®, 2012.
[2] Vahid Khatibi, Dayang N. A. Jawawi, " Software Cost Estimation Methods: A Review", Volume 2 No. 1, Journal of Emerging Trends in Computing and Information Sciences, 2011.
[3] Sweta Kumari , Shashank Pushkar, "Performance Analysis of the Software Cost Estimation Methods: A Review ", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 7, July 2013.
[4] Astha Dhiman, Chander Diwaker, "Optimization of COCOMO II Effort Estimation using Genetic Algorithm", American International Journal of Research in Science, Technology, Engineering & Mathematics, 2013,
[5] Nasa 93 dataset contains effort and defect information available at http://promisedata.googlecode.com/svn/trunk/effort/nasa93-dem/nasa93-dem.arff.
[6] Hamed Shah-Hosseini, "The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm", Int. J. Bio-Inspired Computation, Vol. 1, Nos. 1/2, 2009.
[7] Priti Aggarwal, Jaspreet Kaur Sidhu, Harish Kundra,"applications of intelligent water drops" , IISRO, Multi-Conference, Bangkok, 2013 .
[8] Hamed Shah-Hosseini, "An approach to continuous optimization by the Intelligent Water Drops algorithm", 4th International Conference of Cognitive Science (ICCS 2011)
[9] Mrinal Singh Rawat, Sanjay Kumar Dubey, "Software Defect Prediction Models for Quality Improvement: A Literature Study", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 2, September 2012.
[10] Sunita Chulani, "results of Delphi for the defects introduction model (sub-model of the cost/quality model extension to COCOMO II) ", Center for software engineering, 1997.
[11] Sunita Chulani and Barry Boehm, "Modeling Software Defect Introduction and Removal: COQUALMO (Constructive Quality Model) ", USC - Center for Software Engineering, Los Angeles, CA 90089-0781, 1999.
[12] Capers Jones, "software defect origin and removal methods", Draft 5.0 , December 28, 2012.