Journal of Electronics & Computer Science, Vol.12, No.1, 2010

Development of a Software Effort Estimation Model Using Differential Evolution

Sultan Aljahdali

College of Computers and Information System Taif Universiy ,Taif, Saudi Arabia
aljahdali@tu.edu.sa

Abstract: Effort estimation concerns the issue of computing the number of worker required, the number of hours required to develop a software project. The value of person month normally defined as the effort of a project. The effort is most likely to be the dominate variable for computing the cost and time of a project. Being able to compute the effort at an early stage of the development presents a challenge for software project manager. A powerful algorithm which can mathematical model of the software effort is urgently needed. In [1] author presented number of software effort estimation model based Genetic Algorithms (GAs). They were a modification to the famous Constructive Cost Model (COCOMO). In this paper, we explore the use of Differential Evolution (DE) as an alternative technique to estimate the COCOMO model parameters. The performance of the developed models was tested on NASA software project dataset [2]. The developed COCOMO-DE model was able to provide good estimation for the software effort.

1. INTRODUCTION

The dimension and complication of computer based systems grown noticeably during the past few decades [3]–[6] and the tendency will certainly continue in the future. The cost of developing software is growing and the software is becoming the major cost of the system. Some NASA and Air Force projects have estimated that the cost of software development could be up to 50% of their development cost. The reason is many NASA software projects are considered highly complex system with both hardware/software. For example, the NASA Space Shuttle flies with approximately 500 thousand lines of software code on board and approximately 3.5 million lines of code in ground control station. According to Dr. Patricia Sanders, the Director of Test Systems Engineering and Evaluation at OUSD, in her 1998 Software Technology Conference keynote address,"40% of the DoD's software development costs are spent on reworking the software, which on the year 2000 equal to an annual loss of $18 billion". Furthermore, Sanders stated that only 16% of software development would finish on time and within budget.

Although many research papers appears since 1960 providing numerous models to help in computing the effort/cost for software projects, being able to provide accurate effort/cost estimation is still a challenge for many reasons. They include: 1) the uncertainty in collected measurement, 2) the estimation methods used which might have many drawbacks and 3) the cost drivers to be considered along with the development environment which might not be clearly specified. In this paper, we provide a detailed study on the use of

Differential Evolution is an optimization algorithm which can be used to tune the COCOMO model parameters such that a better effort estimate can be provided [9]. The performance of the developed model was tested on NASA software project dataset provided in [2] and compared to the models presented in [7]. The developed models were able to provide good estimation capabilities compared to other models provided in the literature [1], [7].

2. RELATED WORK

In the past, Soft Computing techniques were explored to build efficient effort estimation models structures [8], [9]. In [10], author explored the use of Neural Networks (NNs), Genetic Algorithms (GAs) and Genetic Programming (GP) to provide a methodology for software cost estimation. Later authors in [11], provided a detailed study on using Genetic Programming (GP), Neural Network (NN) and Linear Regression (LR) in solving the software project estimation.

Many data sets provided in [12], [13] were explored with promising results. In [14], authors provided a survey on the cost estimation models using artificial neural networks. Fuzzy logic and neural networks were used for software engineering project management in [15]. A fuzzy COCOMO model was developed in [8].

Recently, many questions were introduced about the applicability of using Soft Computing and Machine Learning Techniques to solve the effort and cost estimation problem for software systems. In [1], author provided an innovative set of models modified from the famous COCOMO model with interesting results. Later on, many authors explored the same idea with some modification [16]–[19] and provided a comparison to the work presented in [1]. Exploration of the advantages of Fuzzy Logic using the Takagi-Sugeno (TS) technique on building a set of linear models over the domain of possible software Kilo Line Of Code (KLOC) were investigated in [20]. Authors in [7] presented an extended work on the use of Soft Computing Techniques to build a suitable model structure to utilize improved estimations of software effort for NASA software projects. On doing this, Particle Swarm Optimization (PSO) was used to tune the parameters of the COCOMO model. The performance of the developed model was evaluated using NASA software projects data set [2]. A comparison between COCOMO-PSO, Fuzzy Logic (FL), Halstead, Walston-Felix, Bailey-Basili and Doty models were provided with excellent modeling results.

3. SOFTWARE COST ESTIMATION MODELS

A project manager needs to clearly identify the cost estimate of software development so that he/she can evaluate the project progress against expected budget, expected schedule and potentially improve resource utilization in [3]. It was found that the main cost driver for software development is the effort, where effort is translated into cost. The primary element which affects the effort estimation is the developed kilo line of code (KLOC). The KLOC include all program instructions and formal statements [21].

3.1 The COnstructive COst MOdel (COCOMO)

Many software cost estimation models where proposed to help in providing a high quality estimate to assist project manager in making accurate decision about their projects [4, 5]. A well known mathematical model for software cost estimation is the COCOMO model. COCOMO model was first provided by Boehm [4, 5]. This model was built based on 63 software projects. The model helps is defining mathematical equations that identify the developed time, the effort and the maintenance effort. COCOMO model is used to make estimates based upon three different software project estimates.

The three ways of estimating software project effort/cost with increasing levels of accuracy are simple, intermediate and complex models. These three models are defined using increasingly detailed mathematical relationship between the developed time, the effort and the maintenance effort [14]. The estimation accuracy is significantly improved when adopting models such as the Intermediate and Complex COCOMO models [4]. The COCOMO model has the form given in Equation 1.

$$E = a(KLOC)^b \quad (1)$$

E presents the software effort computed in man-months. The values of the parameters a and b depend mainly on the class of software project. Software projects were classified based on the complexity of the project into three categories. They are:

- Organic
- Semidetached
- Embedded.

These models exhibit some nonlinearity characteristics. Extensions of COCOMO, such as COMCOMO II, can be found [6], however, for the purpose of research reported, in this paper, the basic COMCOMO model is used. The three models are given in Table 1. These models are expected to give different results according to the type of software projects [3] (i.e. Organic, semi-detached and embedded) [4, 5].

Table 1: Basic COCOMO Models

Model name	Effort (E)	Time (D)
Organic Model	$E = 2.4(KLOC)^{1.05}$	$D = 2.5(E)^{0.38}$
Semi-Detached Model	$E = 3(KLOC)^{1.12}$	$D = 2.5(E)^{0.35}$
Embedded Model	$E = 3.6(KLOC)^{1.20}$	$D = 2.5(E)^{0.32}$

3.2 Other Software Effort Estimation Models

Typical models for software effort estimation are given in Table 2. These models have been derived by studying large number of completed software projects from various organizations and applications to explore how project sizes mapped into project effort.

Table 2: Known Effort Estimation Models

Model name	Model equation
Halstead	$E = 5.2(KLOC)^{1.50}$
Walston-Felix	$E = 0.7(KLOC)^{0.91}$
Bailey-Basili	$E = 5.5 + 0.73(KLOC)^{1.16}$
Doty (for KLOC > 9)	$E = 5.288(KLOC)^{1.047}$

4. Differential Evolution

Differential Evolution (DE) was first presented in [23, 31] and has proven to be a promising evolutionary process for nonlinear function optimization. DE is very simple technique, fast to converge and easy to implement due to the limited number of control parameters. DE algorithm is similar to Genetic Algorithms (GAs) since they are a population based approach; it also uses similar operators such as crossover, mutation and selection [23]. The main difference between DE and GAs is that genetic algorithms consider crossover as the main operator and mutation as the background operation; while DE relies on mutation operation. The mutation operator in DE is generated using randomly sampled vectors of solutions in the population and combined them to generate the mutant vector using a selected scheme. The evolutionary process is directed by using a selection mechanism. In [8], author presented a comprehensive study of the state of the art on DE, directions for future research and a Matlab toolbox for Differential Evolution. The latest DE-based optimization software is avaliable in several programming languages (C, C++, Matlab, Mathematica, Java, Fortran90, Scilab, Labview) and can be found in [23].

4.1 Population

The initial population for the DE algorithm is more likely to be generated randomly such as in the case of GAs, unless there is a priori knowledge about the problem under study. Assuming the variables to be optimized is defines as $X = \{x_1, x_2, ..., x_n\}$ where $x_i^{low} \le x_i \le x_i^{high}$. Thus, the initial population a_0 can be defined for $i = 1,2,...,n$ as given in Equation 2.

$$a_0 = rand\ [0,1](x_i^{high} - x_i^{low}) + x_i^{low} \qquad (2)$$

In [16], authors investigated the effect of population size on the quality of solutions and the computational effort required by the DE algorithm. They claim that there is a significant influence of the population size on the performance of DE as well as interactions between mutation strategies, population size and dimensionality of the problems. This is an issue we plan to investigate in this paper.

4.2 Mutation

DE utilize a parameter vectors known as $x_{i,G}$, where $i = 1,2,..., NP$, while G stands for the generation number. A mutant vector can be generated according to Equation 3.

$$v_{i,G+1} = x_{r1,G} + F.(x_{r2,G} - x_{r3,G}) \qquad (3)$$

$x_{r1,G}, x_{r2,G}$ and $x_{r3,G}$ are three distinct vectors selected randomly from population G. Note that the selected vectors should be different from each other and also different from the current index point $x_{i,G}$. Thus, the number of parameter vector in a population must be at least four. F is a scaling factor, known as the step size. It is selected in the domain (0, 2]. F controls the amplification of the difference vector $(x_{r2,G} - x_{r3,G})$. Thus, if the population gets close to the optimal F should be decreased. The step-sizes need to be self-adapt over time according to the location of the population of individuals in the search space. This helps producing efficient search to find the optimal solutions [12].

4.3 Crossover

Crossover helps increasing the diversity in the

population. This is similar to the process of crossover in GAs. CR is the crossover rate which is defined in the domain of [0;1). For each target vector $x_{i,G}$, a crossover vector $\{u_{i1,G+1}, u_{2i,G+1}, ..., u_{Di,G+1}\}$ is produced as given in Equation 4.

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & if\ (r_j \le CR)\ or\ j = m, \\ x_{ji,G+1} & if\ (r_j > CR)\ and\ j \ne m \end{cases} \quad (4)$$

where $j = 1,2,..., D$; and $r_j \in [0,1]$ is a random number uniformly distributed; and $m \in (1,2,..., D)$ is the randomly chosen index which ensures that $u_{i,G+1}$ gets at least one element from $v_{i,G+1}$ [10].

4.4 Selection

New individuals are selected for the next population if and only if they achieve a better value for the desired fitness value. The selection mechanism of DE can be presented as in Equation 5, assuming we are minimizing a function f.

$$u_{i,G+1} = \begin{cases} v_{i,G+1} & if\ (f(u_{i,G+1} \le f(x_{i,G}) \\ x_{i,G} & otherwise \end{cases} \quad (5)$$

4.5 Strategies

Different strategies can be adopted in DE algorithm depending upon the type of problem for which DE is applied. The strategies can vary based on the vector to be perturbed, number of difference vectors considered for perturbation, and finally the type of crossover used [31].

The generally used notation for DE is DE/x/y/z. DE stands for Differential Evolution, x represents a string denoting the vector to be perturbed, y is the number of difference vectors considered for perturbation of x, and z denotes the crossover scheme.

The strategy to be adopted for each problem is to be determined separately by trial and error. A strategy that works out to be the best for a given problem may not work well when applied for a different problem [32].

5. Evaluation Criteria

In order to check the performance of the developed model, two evaluation criteria will be adopted.

• we will compute the Variance-Accounted-For (VAF) performance criterion to measure how close the measured values to the values developed using the fuzzy models. Given that $E, \hat{E}$ are the actual effort and the estimated effort, respectively. The VAF is computed as follows:

$$VAF = [1 - \frac{\text{var}(E^{actual} - E^{estimated}}{\text{var}(E^{actual})}] \times 100\ \% \quad (6)$$

• In [30], authors provided an empirical study for data modeling in software engineering application and used Radial Basis Function (RBF) to develop effort estimation model. They considered the Mean Magnitude of Relative Error (MMRE) as the main performance measure. MMRE is defined as:

$$MMRE = \frac{1}{N}\sum_{i=1}^{N}\frac{|E^{actual} - E^{estimated}|}{E^{actual}} \quad (7)$$

We will also adopt these two criteria's for evaluating the cost estimation models investigated here.

6. Experimental Results

Experiments have been conducted on a data set presented by Bailey and Basili [2] to explore strengthen of the developed DE based model. The dataset consist of the following variables:

- Kilo Line of Code (KLOC),
- Methodology (ME)
- Effort (E).

KLOC is described in Kilo Line of code (KLOC) and the (E) is in man-months. The dataset is presented in Table 3. The data was split to two sets training (i.e. 13 projects) and testing/validation (i.e. 5 projects). We used the Matlab Differential Evolution Toolbox [8] to produce our results.

Table 3: NASA Software Project Data

Project No.	KLOC	ME	Measured Effort (E)
1	90.2000	30.0000	115.8000
2	46.2000	20.0000	96.0000
3	46.5000	19.0000	79.0000
4	54.5000	20.0000	90.8000
5	31.1000	35.0000	39.6000
6	67.5000	29.0000	98.4000
7	12.8000	26.0000	18.9000
8	10.5000	34.0000	10.3000
9	21.5000	31.0000	28.5000

10	3.1000	26.0000	7.0000
11	4.2000	19.0000	9.0000
12	7.8000	31.0000	7.3000
13	2.1000	28.0000	5.0000
14	5.0000	29.0000	8.4000
15	78.6000	35.0000	98.7000
16	9.7000	27.0000	15.6000
17	12.5000	27.0000	23.9000
18	100.8000	34.0000	138.3000

7. COCOMO-DE Effort Model based KLOC

In this paper, we plan to use DE to estimate COCOMO model parameters. This will allow us to compute the effort developed for the NASA software projects. The estimated parameters will significantly generalize the computation of the developed effort for all projects.

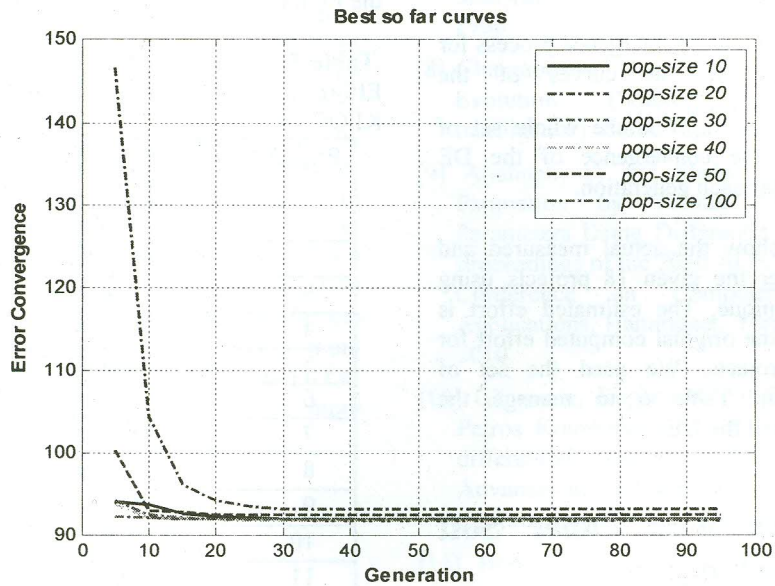

Figure 1: Best so far curves with various population sizes

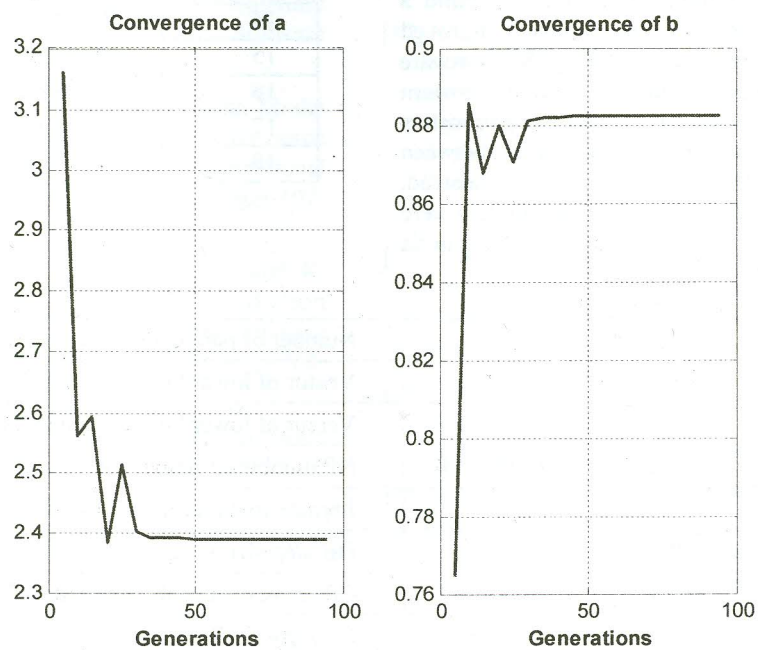

Figure 2: Convergence of the parameters a and b with population size 20

We run DE to estimate the parameters of the COCOMO model presented in Equation 1. In [16], authors pointed that the selection of appropriate population size has a significant effect on the evolutionary process based DE. We computed the values of the COCOMO model parameters a and b using DE with various population sizes. The model with optimal set of parameters is presented in Equation 8.

$$Effort = 2.4649(KLOC)^{0.8781} \quad (8)$$

Figures 1 and 2 show the convergence process for DE (i.e. the best so far curves of the $\sqrt{\frac{1}{N}\sum (E_{actual} - E_{Estimate})}$, N is the whole set of measurements, and the convergence of the DE model parameters after each generation.

In Table 5, we show the actual measured and estimated effort over the given 18 projects using COCOMO-DE technique. The estimated effort is similar in values to the original computed effort for the real NASA projects. We used the set of parameters given in Table 6 to manage the evolutionary process of the DE.

8. Comparison with Other Soft Computing Techniques

In [29], authors presented an extended work on the use of Soft Computing Techniques to build a suitable model structure to utilize improved estimations of software effort for NASA software projects. On doing this, Particle Swarm Optimization (PSO) was used to tune the parameters of the COCOMO model. A comparison between COCOMO-PSO, Fuzzy Logic (FL), Halstead, Walston-Felix, Bailey-Basili and Doty models were provided. In Table 7, we show the MMRE criteria computed over all data set.

It can be seen that the COCOMO-DE model and the COCOMO-PSO models have the same performance. They also found the same optimal set of parameters as described in Equation 8. They outperform the Halstead, Walston-Felix, Bailey-Basili and Doty models. It is also shown that the FL model is the model which provided the minimum MMRE since there are three models with various membership functions. This gives an advantage of the FL model over other effort estimation models.

Table 5: COCOMO-DE: Measured and Estimated Effort in both Training/Testing Cases based the KLOC

Project No.	Measured Effort	DE Estimated Effort
1	115.8000	118.3116
2	96.0000	67.6772
3	79.0000	68.0439
4	90.8000	77.6870
5	39.6000	48.6343
6	98.4000	92.8783
7	18.9000	23.1770
8	10.3000	19.6445
9	28.5000	35.7351
10	7.0000	7.0942
11	9.0000	9.1414
12	7.3000	15.3272
13	5.0000	5.1249
14	8.4000	10.5737
15	98.7000	105.4666
16	15.6000	18.3868
17	23.9000	22.7226
18	138.3000	129.8115

Table 6: Tuning parameters for the DE

D	2	Number of parameters of the objective function
Domain for a	1:10	Vector of lower higher bounds for a
Domain for b	0:1	Vector of lower higher bounds for b
NP	20, 30, 40, 50, 100, 200	NP number of population members
itermax	100	Itermax maximum number of iterations (generations)
F	0.3	DE-step size F ex [0, 2]
CR	0.8	CR: crossover prob. constant ex [0, 1]
Strategy	7	Strategy no. 7

Table 7: The computed MMRE criterion for all models

Model Name	COCOMO based DE	COCOMO based PSO	Fuzzy Model	Halstead Model	Walston Felix Model	Bailey-Basili Model	Doty Model
MMRE	0.0074	0.0074	0.0046	0.1479	0.0822	0.0095	0.1848

9. Conclusions and Future Work

In this paper we proposed a new model structure to estimate the software effort for projects sponsored by NASA using differential evolution. The performance of the developed model was tested on NASA software projects data presented in [2]. The developed COCOMO-DE model was capable of providing good effort estimation of compared to other known model in the literature such as Halstead, Walston-Felix, Bailey-Basili and Doty models. We suggest the use of Genetic Programming (GP) technique to build suitable model structure for the software effort estimation. GP can find a more advanced mathematical function for both the DLOC and ME such that the computed effort is more accurate.

References

[1] Albrecht, A. J. and J. R. Gaffney. Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. IEEE Trans. Software Engineering, 9(6):630-648, 1983.

[2] Bailey, J. W. and V. R. Basili. A Meta Model for Software Development Resource Expenditure. Proceedings of the International Conference on Software Engineering, pages 107-115, 1981.

[3] Benediktsson, Oddur and D. Dalcher and K. Reed and M. Woodman. COCOMO based effort estimation for iterative and incremental software development. Software Quality Journal, 11:265-281, 2003.

[4] Boehm, B. Cost Models for Future Software Life Cycle Process: COCOMO2. Annals of Software Engineering, 1995.

[5] Boehm, B. Software Engineering Economics. Englewood Cliffs, NJ, Prentice-Hall, 1981.

[6] Boehm, Barry and et all. Software Cost Estimation with COCOMO II. Prentice Hall PTR, 2000.

[7] Boraso, M and C. Montangero and H. Sedehi. Software Cost Estimation: An experimental Study of Model Performances. Technical report, 1996.

[8] Chakraborty, Uday. Advances in Differential Evolution (Studies in Computational Intelligence). Berlin: Springer, 2008.

[9] Aljahdali, S. and Sheta, A. Software Effort Estimation by Tuning COOCMO Model Parameters Using Differential Evolution. In the Proceedings of the 2010 ACS/IEEE International Conference on Computer Systems and Applications, Hammanet, Tunisai, May 16-19th, 2010

[10] Gämperle, Roger and Sibylle D. Müller and Petros Koumoutsakos. A parameter study for differential evolution. WSEAS Int. Conf. on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, 2002.

[11] Hodgkinson, A. C. and P. W. Garratt. A Neuro-Fuzzy Cost Estimator. Proceedings of the Third Conference on Software Engineering and Applications, pages 401-406, 1999.

[12] Iorio,, Antony W. and Li,, Xiaodong. Incorporating directional information within a differential evolution algorithm for multi-objective optimization. Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06), pages 691--698, New York, NY, USA, 2006. ACM.

[13] Kelly, Michael A. A Methodlolgy for Software Cost Estimation Using Machine Learning Techniques. Master's thesis, Naval Postgratuate School, Monterey, California, 1993.

[14] Kemere, C. F. An Empirical Validation of Software Cost Estimation Models. Communication ACM, 30:416-429, 1987.

[15] Kumar, S. and B. A. Krishna and P.S. Satsangi. Fuzzy Systems and Neural Networks in Software Engineering Project Management. Journal of Applied Intelligence, 4:31-52, 1994.

[16] Mallipeddi, R. and P. N. Suganthan. Empirical Study on the Effect of Population Size on Differential Evolution Algorithm. Proceedings

of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008), World Congress on Computational Intelligence, June 1-6, Hong Kong, pages 3664-3671, 2008.

[17] Matson, J. E. and B. E. Barret and J. M. Mellinchamp. Software Developemnet Cost Estimation Using Function Points. IEEE Trans. Software Engineering, 20(4):275-287, 1994.

[18] Menzies, Tim and Dan Port and Zhihao Chen and Jairus Hihn and Sherry Stukes. Validation methods for calibrating software effort models. Proceedings of the 27th international conference on Software Engineering (ICSE'05), pages 587--595, New York, NY, USA, 2005. ACM Press.

[19] Mittal, Harish and Pradeep Bhatia. A comparative study of conventional effort estimation and fuzzy effort estimation based on Triangular Fuzzy Numbers. International Journal of Computer Science and Security, 1(4):36-47, 2007.

[20] Mittal, Harish and Pradeep Bhatia. Optimization Criteria for Effort Estimation using Fuzzy Technique. CLEI ELECTRONIC JOURNAL, 10(1):1-11, 2007.

[21] Park R, W. Goethert, J. Webb. Software Cost and Schedule Estimating: A Process Improvement Initiative. Technical report, 1994.

[22] Pillai, K. and S. Nair. A Model for Software Development Effort and Cost Estimation. IEEE Trans. on Software Engineering, 23:485-497, 1997.

[23] Price, Kenneth V. and Rainer M. Storn and Jouni A. Lampinen. Differential Evolution: a Practical Approach to Global Optimization. Berlin: Springer, 2004.

[24] Ryder, J. Fuzzy COCOMO: Software Cost Estimation. PhD thesis, Binghamton University, 1995.

[25] Sandhu, Parvinder S. and Manisha Prashar and Pourush Bassi and Atul Bisht. A Model for Estimation of Efforts in Development of Software Systems. World Academy of Science, Engineering and Technology, pages 148-152, 2009.

[26] Shepper, M. and C. Schofield. Estimating Software Project Effort using Analogies. IEEE Tran. Software Engineering, 23:736-743, 1997.

[27] Sheta, A. F. Estimation of the COCOMO Model Parameters Using Genetic Algorithms for NASA Software Projects. Journal of Computer Science, 2(2):118-123, 2006.

[28] Sheta, Alaa. Software Effort Estimation and Stock Market Prediction Using Takagi-Sugeno Fuzzy Models. Proceedings of the 2006 IEEE Fuzzy Logic Conference, Sheraton, Vancouver Wall Centre, Vancouver, BC, Canada, July 16-21, pages 579-586, 2006.

[29] Sheta, Alaa and David Rine and Aladdin Ayesh. Development of Software Effort and Schedule Estimation Models Using Soft Computing Techniques. Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE CEC 2008) within the 2008 IEEE World Congress on Computational Intelligence (WCCI 2008), Hong Kong, 1-6 June, pages 1283-1289, 2008.

[30] Miyoung Shin and Goel, Amirt L. Emprirical Data Modeling in software Engineering Using Radial Basis Functions. IEEE Trans. on Software Engineering, :567-576, 2000.

[31] Storn, R. On the Usage of Differential Evolution for Function Optimization. Proceedings of the 1996 Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS 1996), pages 519-523, 1996.

[32] Storn, Rainer. Differential evolution design of an IIR filter. Proceedings of the IEEE International Conference on Evolutionary Computation, pages 268--273, 1996. IEEE Press.

[33] Uysal, Mitat. Estimation of the Effort Component of the Software Projects Using Simulated Annealing Algorithm. World Academy of Science, Engineering and Technology, pages 258-261, 2008.

Sultan Hamadi Aljahdali, Ph.D. received the B.S from Winona State University, Winona, Minnesota in 1992, and M.S. with honor from Minnesota State University, Mankato, Minnesota, 1996, and Ph.D. Information Technology from George Mason University, Fairfax, Virginia, U.S.A, 2003. He is the dean of the college of computers and information systems at Taif University. His research interest includes software testing, developing software reliability models, soft computing for software engineering, computer security, reverse engineering, and medical imaging. He is also a member of ACM, IEEE, and ISCA.

Journal of Electronics & Computer Science, Vol.12, No.1, 2010

Consistent Execution of Concurrent Transactions in Peer Data Sharing Systems

Md Mehedi Masud, Sultan Aljahdali

Department of Computer Science, Taif University, Taif, Saudi Arabia.
{mmasud, aljahdali}@tu.edu.sa

Abstract: In this paper, we investigate the mechanisms of transaction processing in a peer data sharing system. A peer data sharing system is a collection of autonomous data sources, called peers, where each peer augments a conventional database management system with an interoperability layer (i.e. mappings) for sharing data and services. In this network, each peer independently manages its database and can execute queries as well as updates over the related data in other peers. For transaction processing mechanism, this paper focuses on the correct execution of concurrent transactions in a peer data sharing system. A correctness criterion is introduced to ensure the consistent execution of concurrent transactions in the system. In order to guarantee correctness, the paper proposes an approach, called Enforced Serialization Order.

Keywords: database, transaction processing, peer to peer

1. Introduction

In a P2P system, all participating computers (or peers) have compatible capabilities and responsibilities, and exchange resources and services through pair-wise communication, thus eliminating the need for centralized servers. Until now, there are many domain specific P2P systems (e.g. Freenet, Gnutella, SETI$@$home, ICQ, etc.) that have already been deployed. With a few notable exceptions, currently implemented P2P systems lack data management capabilities that are typically found in a database management system (DBMS), such as query and transaction processing.

A peer data sharing system (PDS) combines both P2P and database management system functionalities. The local databases on peers are called *peer databases*. In a PDS, each peer chooses its own data model and schemas, and maintains data independently of other peers. Contrary to the traditional data integration systems where a global mediated schema is required for data exchange, in a PDS, the semantic relationships exist between a pair of peers, or among a small set of peers. Any peer can contribute new data, schema information, or schema mappings with other peers' schemas. The data can be shared globally among the peers by traversing the transitive relationships among semantically related peers.

In the last few years, steady progress has been made in research on various issues related to PDSs, such as data integration model, mediation methods [1, 2] coordination mechanisms [3, 4, 5] and data-level mappings [6] among databases in peers. However, the vast majority of the literature on PDSs has focused on query processing, since, intuitively, data sharing in a PDS occurs through query translation and propagation across the network.

In this paper we investigate the execution of transactions in a data sharing system. We observe that transaction processing in a data sharing system is similar to processing in a multidatabase system (MDBS) [7, 8] in the sense that each system consists of a collection of independently created local database systems (LDBSs), each of which may follow different concurrency control mechanisms. Moreover, each system supports the management of transactions at both local and global levels. Global transactions are those that execute at several sites and local transactions are those that execute at a single site. In an MDBS, global level transactions