PREDICTING THE COST ESTIMATION OF SOFTWARE PROJECTS USING CASE-BASED REASONING

Hassan Y. A. Abu Tair

Department of Computer Science
College of Computer and Information Sciences
King Saud University

habutair@gmail.com

Abstract

Many models to estimate the cost of software projects were devised during the last three decades [1], [3]. Although until now there is no perfect model that can estimate the cost precisely, the estimations varies to be *over-estimated* that may lead to using the resources inefficiently and losing a lot of business opportunities or *under-estimated* that may lead to delaying the final software product delivery, project management problems arises, unexpected increase in budget, and low quality of software projects [2]. As a result no accurate decision can be made due to the lack of consistency, this makes the senior project managers depend on their experience to reach the final decision to proceed or cancel the project [4]. In this paper, we apply the artificial intelligence methodology Case-Based Reasoning (CBR) upon an open source software projects dataset in order to assist the project managers to grasp the appropriate cost estimation of a software project.

Keywords -Estimation by analogy, case-based reasoning, and software cost estimation.

1 INTRODUCTION

An expert project manager, depending on his expertise can make a decision of a current project regarding cost estimation to some extent with uncertainty, and as a human he cannot do an intensive computations without a tool that can support his decision, and there is no standard method for expert project manager opinion-based prediction nor accurately he can identify an appropriate estimation for a new project [14]. In 2009, Magne and Stein built up a new model called BEST to assist the project manager in his judgment regarding effort estimation of software projects [2]. A formal and a systematic approach to the project manager judgment is a software estimation by analogy in which a direct comparison of a current project with more historical projects, furthermore expert opinion-based estimation is a human-intensive approach depends on the personal experience of the project manager while the analogy-based system is a data-intensive approach depends on analogous historical projects [14]. CBR systems is acting as project managers experts when they applying analogical reasoning for making an estimation or prediction [4].

In contrast, algorithmic based approaches like COCOMO [3] that depends on regression does not outperform the non-algorithmic analogy based approach [15]. In [14], an empirical evaluation to the analogy based approaches and regression based approaches were conducted using 9 different datasets, the results demonstrate the prediction power of the analogy based system over the regression approaches in all sets, that's does not mean the estimation by algorithmic approaches is rejected, but searching for additional and more accurate methods of software project effort prediction still running in spite of the estimation by analogy appears to be more accurate until now [18]. In analogy based systems, the similarity between two cases depends on the Euclidean distance between the correspondence features, the smaller the distance the more similar it is. In algorithmic models the cost estimation depends on mathematical models as a function of a number of variables, and can be calculated from the formula $\mathrm{Effort} = f(x_1, x_2, \ldots, x_n)$, where *x* is the cost factor (attribute). Each model may have different factors and different function [12].

2 METHODOLGY

In [5], the author described the CBR (Artificial Intelligence methodology) in detail, CBR is uses precedence (previous case or occurrence taken as guidance [6]) to inform current decision. In [14], four stages of a general CBR cycle were described:

1. **RETRIEVE** the most similar cases to the current problem.
2. **RESUSE** the historical cases (previous solutions) to solve the current problem.
3. **REVISE** the proposed solution in order to adapt the current problem.
4. **RETAIN** the approved solution of the current problem in the cases data base for further uses as a historical solution for future problem solving.

In (Fig. 1), A given problem (new project) formulated as a new case (several features describing a problem), CBR system retrieve the most similar cases to a current case then revise the most similar case in order to adapt the needs of the current problem, the revised solution needs testing and approval from the project managers, if it is approved it will be considered as a historical case to be used for later reasoning for new cases.

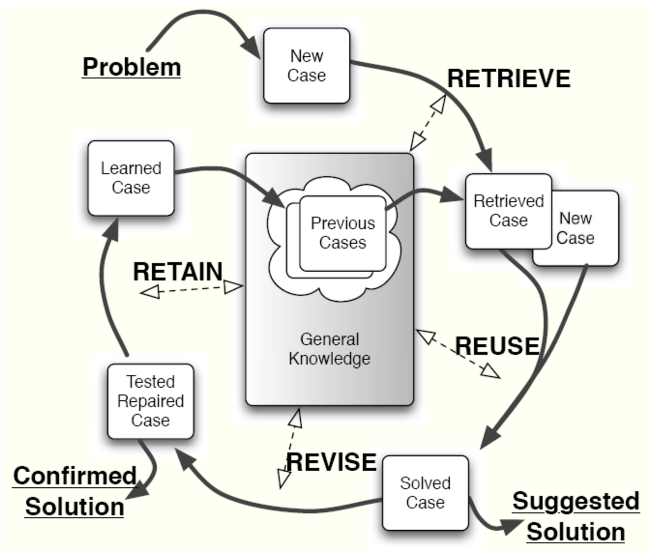

Figure 1: Anatomy of CBR Cycle (Source [14]).

2.1 Why Using CBR?

In [4], the author mentioned several advantages to CBR in which:

1. CBR shows better prediction accuracy than other models according to many studies.
2. CBR is acting as human experts, when they applying analogical reasoning for making estimations.
3. CBR can deal with qualitative and quantitative data.
4. CBR is capable of using an existing solution and revise it to adapt the current problem.
5. It is easy to implement a CBR system.
6. Compared to algorithmic models, CBR shows flexibility and simplicity in use.
7. It is easy to update the CBR data base with a new case; it is a cumulative way of historical cases.
8. CBR is a comprehensive system that encompasses all the software cost estimation steps, retrieve, reuse, revise, and adapt the retrieved case to current case.
9. CBR depends on expert prior knowledge for solving a current problem, which is an advantage.
10. CBR systems have the ability to deal with failed cases.

2.2 Similarity Measures:

Similarity between a current case with different features and other cases (historical cases) in the CBR database depends on a matching function such as k-NN (K-Nearest Neighbor) [4], and already implemented in WEKA tool [11].*k*-NN uses the most common similarity measure in CBR system which is the Euclidean distance metric [14] among cases [15]. Features are a mixture of categorical, discrete, and continuous [7], therefore the Euclidean distance measure is more suitable for features that have a continuous nature; Furthermore k-NN shows the best results in addressing the missing values [14].Suppose that we have two cases with n features, $\mathrm{P} = (\mathrm{p}_1, \mathrm{p}_2, \ldots, \mathrm{p}_n)$ and $\mathrm{Q} = (\mathrm{q}_1, \mathrm{q}_2, \ldots, \mathrm{q}_n)$, then the Euclidean distance without features weights equals:

$\sqrt{\sum_{i=1}^{i=n}(p_i - q_i)^2}$ (1) , and with features weights equals $\sqrt{\sum_{i=1}^{i=n} w_i\,(p_i - q_i)^2}$ (2)

In which the smaller the distance the more similar the two cases are [15]. In [18] the authors state a comprehensive formula (3) and (4) to measure the similarity between two cases for different categories.

$$\mathrm{SIM}(\mathrm{C}_1, \mathrm{C}_2, \mathrm{P}) = \frac{1}{\sqrt{\sum_{1\, \in \mathrm{P}} \mathrm{Feature\_dissimilarity}\left(\mathrm{C}_{1\mathrm{j}}, \mathrm{C}_{2\mathrm{j}}\right)}} \quad (3)$$

Where P is the set of n features, C_1 and C_2 are cases, $\mathrm{C}_{1\mathrm{j}}$ is the feature j of the case C_1, and

$$\mathrm{Feature\_dissimilarity}\ \left(\mathrm{C}_{1\mathrm{j}}, \mathrm{C}_{2\mathrm{j}}\right) \begin{cases} \left(\mathrm{C}_{1\mathrm{j}} - \mathrm{C}_{2\mathrm{j}}\right)^2, & \text{If the features are numeric.} \\ 0, & \text{If the features are categorical and } \mathrm{C}_{1\mathrm{j}} = \mathrm{C}_{2\mathrm{j}}. \\ 1, & \text{If the features are categorical and } \mathrm{C}_{1\mathrm{j}} \neq \mathrm{C}_{2\mathrm{j}}. \end{cases} \quad (4)$$

The main disadvantage of similarity measures is computationally extensive measures, and this can degrade the efficiency of CBR system, but if you deal with less than 100 cases, the efficiency will be not an issue [18].

2.3 Estimation by Analogy:

Estimation by analogy is a type of CBR [18] and analogy in basis is a human basic reasoning process, used by individuals to solve current problems depending on the past similar cases [14] furthermore analogy based estimation is widely used and more accurate than other algorithmic approach. Estimation by analogy fall into three different categories regarding effort estimations in addition to algorithmic models and expert judgment [17], thereby analogy based cost estimation can produce sound estimates in which to be useful and accepted by the practitioner [15]. The performance of analogy depends mainly on the availability of data sets [14], and as mentioned in [15] analogy-based estimation outperforms regression approaches for a number of real-world data sets [14], and the analogy based estimations shows a remarkable performance although many experts have encouraged the practitioners to use more than one method thereby increasing or reducing their confidence in the estimation of software projects [18].

Analogy based estimations is a systematic approach, it follows a number of steps for estimating a software project cost. In [16] the authors summarize these steps which are:

1. Measuring or estimating the values of the software projects (cases) metrics for a current case.
2. Searching the repository for similar case(s) to the current project and select them as analogues.
3. The cases retrieved are considered as initial estimates to the current problem.
4. Comparing the metric values of the current project and retrieved projects (expected solutions).
5. Adjusting and revising the differences between the current project and the most similar one in order to adapt the current case to fulfill its needs.

The adapted case will be saved in CBR database for further reuse in the future as a new historical case, the adaptation process in most cases is a must because each project has its own metrics and scope which is different to some extent of another project.

2.4 ANGEL Tool:

ANGEL is a software tool for estimation by analogy (case-based reasoning) approach for software project cost prediction; it provides a lot of functionalities [8]. ANGEL is the most popular tool in the literature for software estimation by analogy using CBR. In the late 90s, Professor Martin Shepperd led a team of researchers and students to develop the

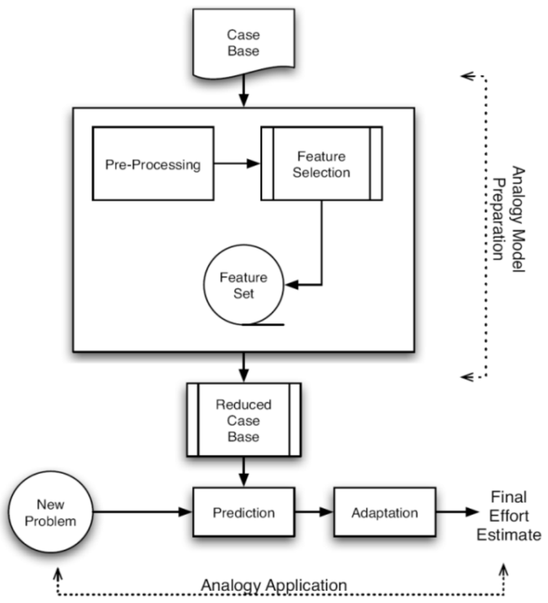

Figure 2: ANGEL application process (Source [14]).

ANGEL tool at the Empirical Software Engineering Research Group (ESERG) at Bournemouth University, UK [9], [14]. Furthermore ANGEL estimation proved to be the most accurate of all methods in different cases [16], and considered the dominant automated software effort estimation [14]; also it is very successful in providing accurate estimates [4]. (Fig. 2) shows the process of ANGEL tool phases through the pre-processing of cases and then features selection to bring up a final features set to be used in the prediction process, the reduced case base is used then for prediction the effort of a new problem, adaptation may be required before final effort estimation.

3 EXPEREMNTAL DATASET

3.1 Dataset:

The dataset used in this paper contains of 126 open software projects cases. It was first extracted by a software metrics tool [13] and built by the authors of [10]. They classified these projects into three different domains of C# projects. Table 1 shows the number of these projects and their percentage according to each domain.

Domain	Number of applications	Percentage
Communication	33	26.2%
Finance	45	35.7%
Game	48	38.1%

Table 1: Dataset domains and percentage (Source [10]).

3.2 Features Selection:

Features selection is very important for improving the accuracy of estimations and minimizing the complexity and time needed to come up with certain estimation [18], thus most of estimation methods depend on the project characteristics and features for deriving cost estimation from the cost drivers [15]. Among of these features the ones used by COCOMO II model (table 2) is a snapshot.

Cost Factor	Description
	Product
RELY	required software reliability
DATA	database size
CPLX	product complexity
	Computer
TIME	execution time constraint
STOR	main storage constraint
VIRT	virtual machine volatility
TURN	computer turnaround time
	Personnel
ACAP	analyst capability
AEXP	application experience
PCAP	programmer capability
VEXP	virtual machine experience
LEXP	language experience
	Project
MODP	modern programming practice
TOOL	software tools
SCED	development schedule

Table 2: The cost factors in COCOMO II model (Source [12]).

Software Metric	Description
Lines	Total lines of Code
LOC	Lines of Codes without comments or empty lines
SLOC	Statements Line of Codes
SLOCmath	Counting all math operators
MCDC	Modified Condition/Decision Coverage
MaxNest	Maximum Nesting
CComplexity	Cyclomatic complexity
AvgMethod	Average Methods per class
MethodCComplexity	Methods Cyclomatic complexity
MaxInheritanceDepth	Maximum inheritance depth
AvgDependency	Average dependency
ChildNumber	average or max number of children per class

Table 3: Software Metrics.

Regarding the CBR system for estimation by analogy, In [4] the author mentioned the most important software cost factors (features) to be considered in CBR method are: (*Project size, Organization type, Target platform, Quality of system requirements, Development type, Business area , Application type, Project security, Complexity of the software, Staff experience, Development environment*).

In contrast in our study we used the software metrics (measurements of the source code of software projects), we used the same open source projects metrics in [10], and these metrics are presented in (table 3). In [10] they used the information gain for subset selection of metrics, in which the metric (feature) that has the highest information gain was considered as the best metric for labeling a tuple in the dataset. In our study we depend on that information gain presented in (table 4) to assign weights to features (metrics). we associated the metrics in the CBR system with a priority level varying (High, Mid, Fair, Low), and respectively associated with numbers varying from 4 down to 1, only for the metrics appeared in (table 4), the rest of metrics in (table 3) will be assigned the lowest priority by default which is 1. For example LOC has been assigned a priority of 3 depending on its information gain value in (table 4) which is 0.336.

Software Metric	Information Gain Value	Software Metric	Information Gain Value
MCDC	0.479	SLOCmath	0.322
CComplexity	0.45	MaxNest	0.299
CBO[1]	0.363	avgMethod per class	0.151
LOC	0.336	Childern	0.147

Table 4: Metrics information gain (source [10]).

3.3 Accuracy and Prediction:

The accuracy of an estimation method is needed, and for each estimation there is a fundamental question pops up "How accurate are the predictions?",therefore the accuracy is defined as the mean magnitude of relative error (MMRE), it is the mean of percentage errors as in equation (5) [18].

$$\mathrm{MMRE} = \sum_{i=1}^{i=n} \left(\frac{|E - \hat{E}|}{E} \right)_{\mathrm{i}} \frac{100}{\mathrm{n}} \qquad (5)$$

Where n is the number of projects (i.e. cases), E is the actual effort, and $\hat{\mathrm{E}}$ is the predicted effort. In [18], the authors mentioned that the MMRE is not always appropriate indicator of the prediction, where the outlier and extreme values can affect the final prediction, so they additionally used the *Pred*$_{25}$ (6) which is "The percentage of predictions that fall within 25 percent of the actual value". In this paper we use both, MMRE and *Pred*$_{25}$ as a performance measures. Those prediction measures are widely used in the literature [18].

$$Pred_{25} = |P|^{-1} \left| \{ p \in P \,\middle|\, |r_p| \leq 0.25 \} \right| \qquad (6)$$

Where P is the number of projects (cases) and $r_p = (\hat{e}_p - e_p)/e_p$, where $\hat{e}_p$ is the predicted effort while e_p is the actual effort [15].

4 RESULTS

As mentioned in (table 1), we have three different domains of projects in the dataset used, in our experiments we used each domain as a different data set for two reasons: **1**) The Authors in [10] shows a variety of aspects regarding the three domains in term of software metrics, for example finance applications require larger number of codes, and game applications have a higher value of object oriented metrics while the communications projects complexity is higher than the fanatical projects. Although these aspects are experimental based point of view, it also depends on the experience of the programmer to some extent. **2**) The efficiency will be not an issue if the number of dataset cases is less than 100 cases [18].

Also we used the information gain values in (table 4) to assign weights to the features in the case template of CBR system to better retrieve the most similar cases to the current project (case). The results of MMRE and *Pred*$_{25}$ shown in (table 5) and (table 6) respectively are tested using the ANGEL analogy based tool, in which the higher Pred$_{25}$ score means better predictive accuracy, and the smaller MMRE means a better predictive accuracy [18].
Our target feature is MCDC because it has the highest information gain value which is 0.479 as shown in (table 4). The Jack knifing validation technique was used in which each case removed from the dataset and the rest used to predict the removed one, after that the removed returned back to the dataset and another one is picked to be predicted until all cases finished [18].

Computing the MMRE and *Pred*$_{25}$ needs a remarkable time, so both of them are considered as computation extensive measures, in our tests, it takes about 4 hours to give the results, it depends on the number of cases being tested.

1. **CBO**: the **Coupling Between Object** Classes.

The more cases you have the more Computations time you need. Also the symbol K shown in the tables below denotes the number of analogues.

Dataset	Dataset size	K=1	K=2	K=3	K=4	K=5
Communication	33	0.346	0.332	0.397	0.386	0.408
Finance	45	0.212	0.22	0.25	0.256	0.247
Game	48	0.2	0.196	0.215	0.234	0.263

Table 5: MMRE measure regarding Dataset Size and Number of Analogies.

Dataset	Dataset size	K=1	K=2	K=3	K=4	K=5
Communication	33	48.485	54.545	60.606	60.606	57.576
Finance	45	73.333	75.556	73.333	77.778	71.111
Game	48	77.083	79.167	72.917	66.667	68.75

Table 6: *Pred*$_{25}$ measure regarding Dataset Size and Number of Analogies.

The presented data in (table 5) and (table 6) have been plotted in (fig. 3) and (fig. 4) respectively. In (fig. 3) and (fig. 4), we can note the game and finance datasets is more related to each other by looking to the convergence of the performance measures plotting.

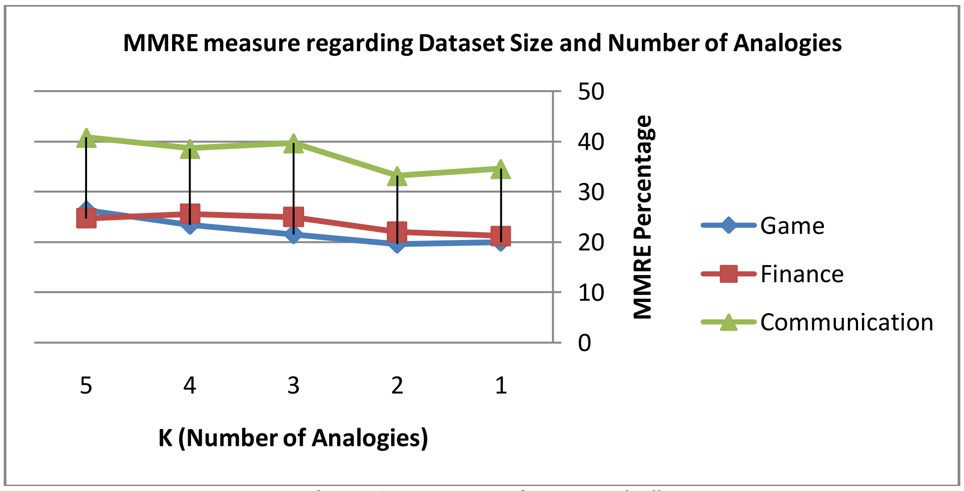

Figure 3: MMRE performance indicator.

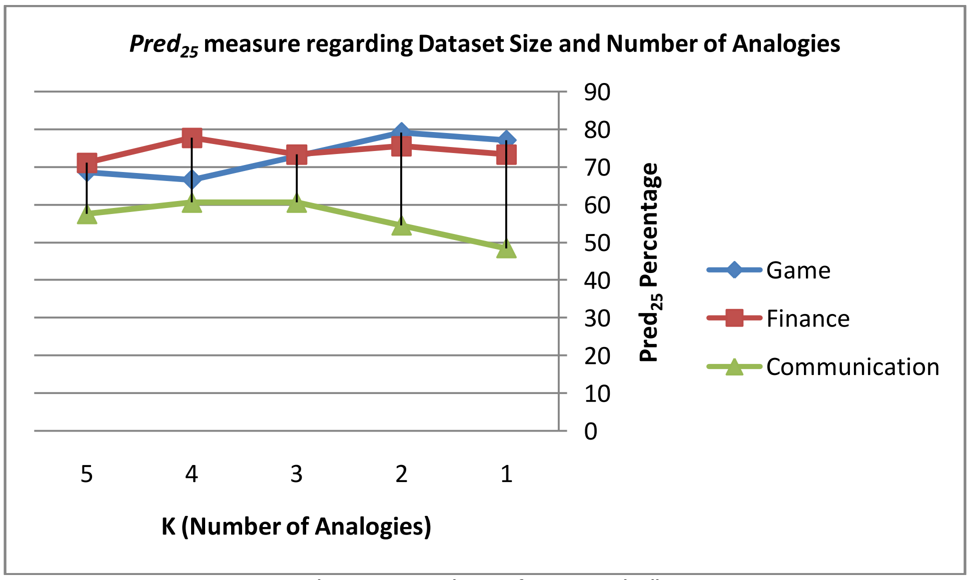

Figure 4: *Pred*$_{25}$ Performance indicator.

Also for MMRE and $Pred_{25}$ measures, the more cases we have the more accurate measures we can get, although we can note that $Pred_{25}$ is more accurate than MMRE, but both of them give an interesting results for the different datasets, the average of MMRE measure for communication, finance, and game datasets are 37.38, 23.7, and 22.16 percent respectively while the average of $Pred_{25}$ measure for communication, finance, and game datasets are 56.3636, 74.222, and 72.9168 percent respectively.

5 DISCUSSION

In our approach, we introduced the cost estimation of software projects by extracting the code metrics, then comparing the metrics of a current project to the historical ones in the database. Specifically, the main purpose of our work is to let the project manager estimates the cost per releases or stages, i.e. by using our approach the practitioners can extract the metrics of code implemented for a certain stage, to predict the cost of the next stage of the same project. This can give the project manager extra knowledge to help taking a proper decision in the stage agreements of a certain projects. Also this can be considered as a support step to the project manager to further estimate the process of development after finishing each release. Even more the project managers can take aware of the risks that may pops up depending on the analysis or comparing the code metrics of different stages. This could be one of the main files that the project manager can pay attention before establishing a new stage. This task can be assigned to the IT project manager to extract features from the implemented code and try to estimate the effort and predict what can be done in the next stage or even according to the information he has got, the Project manager can decide if he can proceed or not, or even escalades any issue appeared to the project sponsor. Also predicting the effort for a project releases or stages is more accurate to the project Manager, because he is dealing with the same team and he knows more about the code in the previous releases. This gives him a good view to decide what can be done later. Furthermore our work is the first study that applies the CBR on open software projects metrics, to predict the cost per releases or stages of a certain project.

6 CONCLUSION

Predicting the cost of software projects is a remarkable challenge to the project managers. The expert managers can depend on their expertise to grasp an appropriate estimation to some extent, and others depend on the algorithmic based models like COCOMO to predict the effort of a certain problem. Most research papers that tackle the non-algorithmic approach like estimation by analogy show the power of analogy based reasoning in predicting the cost of software projects, in which picking the most similar solutions from the case base of historical cases. In our paper we used the software metrics of an open software projects to predict the most similar solution to the current problem, our result shows an interesting achievements of the performance indicators for different number of analogues. Our work can support the project manager decision in order to choose the most appropriate case to his current problem depending on the code metrics.

7 ACKNOWLEGMENT

This work was supported by the Research Center of College of Computer and Information Sciences, King Saud University, and we are grateful for this support.

8 REFERENCES

[1] LH Putnam, A general empirical solution to the macro software sizing and estimating problem. IEEE Trans. on Softw. Eng., Volume 4, No 4, pp 345-61, April 1978.

[2] M. Jørgensen and S. Grimstad. Software Development Effort Estimation: Demystifying and Improving Expert Estimation, In: Simula Research Laboratory - by thinking constantly about it, ed. by AslakTveito, Are Magnus Bruaset, OlavLysne. Springer, Heidelberg, chap. 26, pp. 381-404, 2009.

[3] B. Boehm, B. Clark, E. Horowitz, R. Madachy, R. Shelby, and C. Westland, Cost models for future software life cycle process: COCOMO 2.0 in Annals of Software Engineering Special Volume on Software Process and Product Measurement. J. D. Arther and S. M. Henry, Eds., vol. 1, pp. 45–60, J.C. Baltzer AG, Science Publishers, Amsterdam, The Netherlands, 1995.

[4] Hasan Al-Sakran, Software Cost Estimation Model Based on Integration of Multi-agent in Journal of Computer Science 2 (3): 276-282, 2006.

[5] A. Aamodt, E. Plaza (1994); Case-Based Reasoning: Foundational Issues, Methodological Variations, and System
Approaches.AI Communications. IOS Press, Vol. 7: 1, pp. 39-59.

[6] Collin Dictionary.

[7] Colin Kirsopp, Martin Shepperd, John Hart, Search Heuristics, Case-Based Reasoning and Software Project Effort Prediction, Empirical Software Engineering Research Group School of Design, Engineering and Computing Bournemouth University.

[8] ArchANGEL software tool for project prediction http://dec.bmth.ac.uk/ESERG/ANGEL, last visit October 2012.

[9] Shepperd, Martin, Chris Schofield, and Barbara Kitchenham. "Effort estimation using analogy." Proceedings of the 18th international conference on Software engineering. IEEE Computer Society, 1996.

[10] Hassan Najadat, Izzat Alsmadi, and Yazan Shboul, Predicting Software Projects Cost Estimation Based on Mining Historical Data, International Scholarly Research Network, ISRN Software Engineering, Volume 2012, Article ID 823437, 8 pages.

[11] Weka: A Data Mining Software, http://www.cs.waikato.ac.nz/ml/weka, last visited October 2012.

[12] H. Leung, Z. Fan, Software Cost Estimation, H Leung, Z Fan - Handbook of Software Engineering, Hong Kon, 2002.

[13] I. Alsmadi and K. Magel, “Open source evolution analysis,” in Proceedings of the 22nd IEEE International Conference on Software Maintenance (ICSM ’06), Philadelphia, Pa, USA, 2006.

[14] Keung, Jacky. "Software Development Cost Estimation Using Analogy: A Review." Software Engineering Conference, 2009. ASWEC'09. Australian. IEEE, 2009.

[15] Auer, Martin, et al. "Optimal project feature weights in analogy-based cost estimation: Improvement and limitations." Software Engineering, IEEE Transactions on 32.2 (2006): 83-92

[16] Walkerden, Fiona, and Ross Jeffery. "An empirical study of analogy-based software effort estimation." Empirical Software Engineering 4.2 (1999): 135-158.

[17] Papatheocharous, Efi, Harris Papadopoulos, and Andreas S. Andreou. "Feature Subset Selection for Software Cost Modeling and Estimation." arXiv preprint arXiv:1210.1161 (2012).

[18] Martin Shepperd and Chris Schofield, Estimating Software Project Effort Using Analogies, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 23, NO. 12, NOVEMBER 1997.