Journal of Theoretical and Applied Information Technology
31st May 2014. Vol. 63 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN EFFICIENT APPROACH TO DEVELOP SOFTWARE COST ESTIMATION MODEL USING CASE-BASED REASONING AND AGENT TECHNOLOGY

[1] HASAN AL-SAKRAN, [2] HASSAN Y. A. ABU TAIR
[1]Assoc. Prof., Department of Management Information Systems, King Saud University, Saudi Arabia
[2]Department of Computer Science, King Saud University, King Saud University, Saudi Arabia
E-mail: [1]halsakran@ksu.edu.sa, [2]habutair@gmail.com

ABSTRACT

One of the most important tasks for IT professionals is software development cost estimation. This critical task affects the firm's software investment decisions before bidding for a contract or committing required resources to that project. Under-estimation may lead to unexpected increase in budget, delay of project completion or its low quality, while over-estimation may lead to losing business opportunities. In this work authors investigate possibility of building a software cost estimation model by using multi-agent system to collect software cost data from distributed predefined project cost databases. The developed model implements Case-Based Reasoning (CBR) to find similar projects in historical data retrieved from measured software projects developed by different organizations, which will assist the project managers to perform an appropriate cost estimation of a software project.

Keywords: *Analogy, Case-Based Reasoning, Agent Technology, Magnitude of Relative Error.*

1. INTRODUCTION

Accurate software cost estimation is critical task for every firm's software investment decision. The accuracy of estimation of software project cost has a direct and significant impact on the quality of such decisions. Management carefully considers costs and benefits of software before committing the required resources to that project or bidding for a contract. Sometimes wrong estimation is very high; in fact it could be 150-200% higher than the actual cost [1].

A lot of software projects cost estimation models have been developed during the last four decades [1], [2]. But until now there is no perfect model that can give an estimation close enough to the actual cost. Over-estimation may lead to using the resources inefficiently and losing a lot of business opportunities, while under-estimation may lead to delaying the final software product delivery, unexpected increase in budget, or low quality of software projects. As a result no accurate decision can be made due to the lack of consistency, so the senior project managers become depended on their experience to reach the final decision to proceed or cancel the project.

Using data on projects of single organization for cost estimation has its benefits, such as ease of understanding and controlling collected data. But there have been multiple reports on contradictory results for different software cost estimation modeling techniques using such data. Myrtveit [3] concluded that generalization of the obtained results is difficult due to the characteristics of the datasets being used and their small size. Results of the studies that rely on using organization-specific datasets makes them more biased because the data at hand are specific to a given organization. Characteristics of the datasets being used play a major role.

Boehm [2] and Mendes [4] suggested that relying on organization-specific datasets leads to poor software cost predictions due to the following problems: collection of data on previous projects from single organization could be too expensive; information on older projects may outdated and no longer be valid or appropriate due to the new technologies that organization is using; and difficulty to ensure consistency of the collected data.

The authors focus on usage of a large number of datasets and the issues of the dataset characteristics in very large collected data. Datasets are selected from distributed software project databases of different organizations of comparable domains. Organizations that do not have their own data or

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

expertise to build their own cost estimation models will be able to do so using this approach.

Using artificial intelligence (AI) methodologies in software cost estimation provides consistency and better accuracy of estimation. This paper presents an alternative approach of software cost estimation based on Case Based Reasoning (CBR), similar to the one applied by Shepperd [5], combined with mobile agent technology. CBR makes use of previous experience to solve newly encountered problems. The past experience is recorded as cases in a CBR's case base. When a new problem emerges, the system retrieves projects from the database to find similar cases to the current problem, and the closest match is modified to fit the new problem. New solution will be stored in the case base as a learned case to preserve the experience and can be reused in the future.

In CBR problem-solving is seen as a process, which involves the retrieval of similar prior cases from case bases using mobile agent methodology and the adaptation of retrieved cases' solutions to fit the new problem's requirements. CBR systems is acting as project managers experts when they applying analogical reasoning for making an estimation or prediction.

In analogy based systems, the similarity between two cases depends on the Euclidean distance between the corresponding features. The smaller the distance the more similar they are. In algorithmic models the cost estimation depends on mathematical models as a function of a number of variables, and can be calculated using the formula Effort = $f(x_1, x_2, \ldots, x_n)$, where x is a cost factor (attribute). Each model may have different factors and different functions.

The rest of the paper is organized as follows. Section 2 briefly reviews the major cost estimation models. Sections 3 and 4 describe the major features of CBR and mobile agent technology respectively. Section 5 discusses how to implement mobile agent technology integrated with CBR in order to build the software cost estimation model. Section 6 discusses the experimental data set and the findings. Section 7 summarizes the discussions and suggests direction for future work.

2. RELATED WORK

Some potential solutions of the above problem have been developed based on algorithmic models (e.g. Constructive Cost Model (COCOMO, COCOMO II [6], Function Points, Price-to-Win, and SLIM [7]. Most of the algorithmic software estimation models are based on analytical methods and derived from the statistical or numerical analysis of historical projects data.

The general form of equation used by COCOMO and Function Points methods can be represented as

$$E = xS^y, \quad (1)$$

where E is effort, S is size measured as number of lines of code or function points, x is a productivity parameter and y is economics of scale parameter. COCOMO model provides three equations according to the project development mode (embedded, semi-detached, and organic). Their parameters need to be adjusted to local circumstances.

COCOMO II provides tool for conducting empirical analysis of the model. The general form of equation used by COCOMO II is:

$$E = X \times [S]^{B+\sum SF} \times \prod EM, \quad (2)$$

where X is baseline multiplicative constant, B is baseline exponential constant, SF are scale factors (understanding product objectives, flexibility, team coherence, etc.), EM are effort multipliers (software reliability, database size, reusability, complexity, etc.)

Later, Vinaykumar [8] used wavelet neural networks for the prediction of software cost estimation. Unfortunately the accuracy of these models is not satisfactory so there is always a place for more accurate software cost estimation techniques. Lefley and Shepperd [9] applied genetic programming to improve software cost estimation on public datasets with great success.

Kumari [10] provide a comparative study on support vector regression, Intermediate COCOMO and Multiple Objective Particle Swarm Optimization model for estimation of software project effort, based on the size of the code and set of cost drivers.

The common problem with methods mentioned above - none of them can be considered convincing or consistent in solving the cost estimation problems [11]. Some of these algorithmic methods may lead to relative errors as high as 600% [12].

Due to the shortcomings of the above mentioned methods researchers turn their attention to non-algorithmic methods, and, in particular, to a set of approaches based on expert judgment, neural networks, rule based and case based reasoning.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Expert judgment methods rely on the use of human expertise to estimate software cost [13] and can be used when no quantified empirical data is available. Estimators rely on their past experiences and understanding of the problem. They search for previous cases similar to the new project and adapt old estimations to fit the new situation. This approach however lacks a consistent and systematic procedure for cost estimation and can result in over- or under-estimation of the cost of the software project. The major drawback of this method is that an estimate is only as good as the expert's opinion.

Other AI techniques also have been used for estimation. Kellner [14] developed a rule based system to estimate the cost of software. This technique has been adopted from the AI domain where a known fact fires up rules, which in turn may assert new facts.

A lot of effort has been put into development of neural network based software estimation models [15]. Neural network estimation models have to be trained on historical data from older projects that used as input values (project size, complexity, skill levels, etc). Model's algorithmic parameter values are automatically adjusted until they are acceptable for predicting results for the training data set. As for any algorithmic technique, these models require very large data sets in order to accurately train neural networks.

Jørgensen et al [16] built a new model called BEST to assist project managers in their estimation of effort involved in software projects. An empirical evaluation of the analogy based and regression based approaches were conducted using 9 different datasets [17]. The results demonstrated superiority of the analogy based prediction system over the regression approaches in all sets. That's does not mean the estimation by algorithmic approaches must be rejected, so searching for additional and more accurate methods of software project effort prediction still running even so the estimation by analogy appears to be more accurate now [19].

CBR method is rated among the best methods in a variety of circumstances [20]. Experiments showed that this approach provides better accuracy than algorithmic methods. Algorithmic are capable of handling different types of problems in comparison with CBR where solutions are derived from historical cases and use a form of reasoning close to the human problem solving as opposed to rule based reasoning or neural nets. CBR can when generation of an algorithmic model is difficult or impossible (no statistically significant relationships could be found).

3. MOBILE AGENTS

Mobile agents can be defined as autonomous computational entities capable of effectively performing operations in dynamic environments that are known as multi-agent systems. Agents are capable of exercising control over their actions, and can interact and/or cooperate with other agents.

The proposed system is based on authors' previous works [21], [22]. The system supports multiple intelligent agents that search distributed databases for most similar cases. Agents access case bases to retrieve the best matches. There is no guarantee that any agent may find the best similar case. It will retrieve the most similar local case, and cooperation among agents may lead to achievement of the final goal of finding the best match. This way the cost prediction of a project does not just rely on few projects stored locally, but affected by data retrieved from distributed databases (distributed datasets).

Intelligent agents can possess some or all of the following characteristics: autonomy, mobility, reacting to changes in the environment; ability to cooperate, learn from experience and communicate with other agents.

4. CASE BASED REASONING

Case Based Reasoning (CBR) has been attracting much attention recently as a paradigm with a wide variety of applications. This paper discusses issues related to construction of a cost estimation model and composition of a case, where sub cases are distributed across different distributed databases.

CBR approach is based on re-using past experience or occurrence [23], where a reasoner recalls previous cases similar to the current one and uses them to solve the current problem. The CBR repository stores numerous cases related to the matters under consideration. Often the past experiences provide important clues or direct answers to the current problem.

Aamodt et al [18] described CBR technique as combination of the following four processes and shown on figure 1:

- Retrieve of the most similar previous case related to the current problem.
- Re-use this or these case(s) to solve the current problem.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- Revise the solution based on re-using previous cases in order to adapt to the current problem.
- Retain the new solution (as a new case) in the case-base. Such a way, a CBR system will gradually grow larger and become a precious resource.

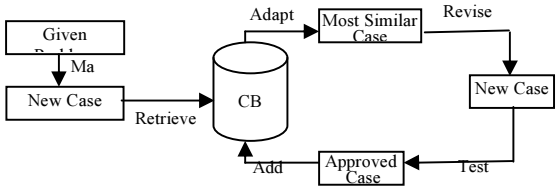

Figure 1: CBR cycle

The adaptation process in most cases is a must because each project has its own metrics and scope. Case-based reasoning has quite a few advantages:

- According to many studies CBR shows better prediction accuracy than other models.
- Applying analogical reasoning for making estimations CBR is acting as human experts.
- CBR can deal with qualitative and quantitative data.
- CBR is capable of using an existing solution and revising it to adapt the current problem.
- Implementation of a CBR system is easy.
- Compared to algorithmic models, CBR shows flexibility and simplicity in use.
- It is easy to update the CBR data base with a new case; it is a cumulative way of historical cases.
- CBR is a comprehensive system that encompasses all the software cost estimation steps, retrieve, reuse, revise, and adapt the retrieved case to current case.
- CBR depends on expert prior knowledge for solving a current problem.
- CBR systems have the ability to deal with failed cases.

Establishing similarity of cases is the basis of CBR and case searching. It depends on a set of attributes which make the case different from others. These attributes are the key attributes of a case. The cases which have one or more similar key attributes are similar. Each key attribute represents a specific characteristic of this case. Examples of the key attributes are project size, target platform, quality of system requirements. Project size, one of the more likely used key attributes that represents the number of lines of code of the project, can be estimated using different techniques such as Genetic Programming and Neural Networks [24] Other attributes, such as development environment, application type, business area type, etc., can act as sub-key attributes.

Case searching model is used to compare and filter the cases from the case base to find the similar ones. It is based on the key and sub-key attributes. To make the case searching model more effective, case index reflecting the main feature of the cases is build. This index is recommended especially when the volume of the case base is large.

4.1 Similarity Measures.

Similarity between a current case with a set of features and other cases (historical cases) in the CBR database depends on a matching function such as k-NN (K-Nearest Neighbor) [21], which already is implemented in WEKA tool [25].The above mentioned function uses the most common similarity measure which is the Euclidean distance metric among cases. Features can be categorical, discrete, and continuous by nature [6], 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 [26]. Assuming there are two cases with n features, P = (p_1, p_2… p_n) and Q = (q_1, q_2…q_n) the Euclidean distance without features' weights can be computed as:

$$\sqrt{\sum_{i=1}^{i=n}(p_i - q_i)^2}, \quad (3)$$

and with features' weights:

$$\sqrt{\sum_{i=1}^{i=n} w_i (p_i - q_i)^2}, \quad (4)$$

where the smaller the distance the more similar the two cases are. In [19] the authors state a formula (5) and (6) to measure the similarity between two cases for different categories.

$$SIM(C_1, C_2, P) = \frac{1}{\sqrt{\sum_{1 \in P} \text{Feature\_dissimilarity}(C_{1j}, C_{2j})}}, \quad (5)$$

where P is the set of n features, C_1 and C_2 are cases, C_{1j} is the feature j of the case C_1, and

$$\text{Feature\_dissimilarity}(C_{1j}, C_{2j}) \begin{cases} (C_{1j} - C_{2j})^2, & \text{If the features are numeric.} \\ 0, & \text{If the features are categorical and } C_{1j} = C_{2j}. \\ 1, & \text{If the features are categorical and } C_{1j} \neq C_{2j}. \end{cases} \quad (6)$$

The main disadvantage of similarity measures is high computational requirements, and this can degrade the efficiency of CBR system, but if you

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

deal with less than 100 cases, the efficiency will be not an issue [19].

4.2 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 [7], [17]. Furthermore ANGEL estimation proved to be one of the most accurate of all methods in different cases, and considered the dominant automated software effort estimation [26]. The main phases of the process of ANGEL tool consist of re-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 of the effort required for a new problem. Adaptation may be required before final effort estimation.

5. INTEGRATION OF CBR AND MOBILE AGENT APPROACHES INTO COST ESTIMATION MODEL

The overall framework of the proposed system is presented in figure 2. It is composed of three different major components: front end user machine, back end server, and the software cost estimation servers on the web. The system has a number of agents. Each agent is designed to represent a specific functional unit. This requires three different agent types, one mobile and two static (interface agent (IA), task agent (TM), and mobile information agent MIA).

A client at the front end user machine communicates with the system through a web browser. The TM agent on the back end server generates multiple MIAs which search the web for the required information. Each agent visits a software cost estimation server on the web. It carries a searching criteria generated at the back end server every time a client conduct search. When an agent finds the most similar according to the similarity metric project will send it back to the task agent at the back end server where it will be filtered and then presented to the user.

The steps of the cost estimation process:

- A client communicates with the system through interface agent. The task agent generates MIA that should do the following:
- Each mobile information agent should select projects of the same types, similar application domains, size, etc. (to be defined by the client) from CBR database.
- The task agent RECEIVE candidate projects from each mobile agent; MERGE candidate projects; and CHOOSE best project(s) using ANGEL tool as explained above.
- When a project is retrieved, its attributes are match against those of the current project. If it is a perfect match then solution is found, else unmatched attributes are extracted and grouped as a new project. These unmatched attributes are used again to retrieve the most similar project, and so on as long as any unmatched attributes remain. The task manager will merge all collected projects. If retrieval procedure has resulted in large number of projects with the same attributes, then similarity function is computed and compared to find the closest match.

6. EXPERIMENTAL DATASET

6.1 Dataset

The dataset used in this work contains 126 software project cases selected from three different application domains (communication, finance, and games) of C# projects. Initially the dataset was extracted by a mobile agent that was injected by the software metrics tool as proposed in [13], [17], [20].

6.2 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, thus most of estimation methods depend on the project characteristics and features for deriving cost estimation from the cost drivers. Among these features are the ones used by COCOMO II model (refer to table 1).

In this study authors used the same software metrics (measurements of the source code of software projects), as in [13]. The description of these metrics presented in table 2. Information gain value has been used for selection of metrics subset, where the metric having the highest information gain was considered the best metric for labeling a tuple in the dataset. This study depends on the information gains presented in table 3 of the metrics used to assign their weights. Authors associate the metrics in the CBR system with a priority levels (High = 4, Mid = 3, Fair = 2, Low = 1) only for the ones that appeared in (Table 3). The rest of metrics in table 2 will be assigned by default the lowest priority which is 1. For example LOC has been assigned a priority of 3 depending on its information gain value table 3 which is 0.336.

Journal of Theoretical and Applied Information Technology
31st May 2014. Vol. 63 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: Cost Factors In COCOMO II Model.

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
	Project
MODP	modern programming practice
TOOL	software tools
SCED	development schedule
	Personnel
ACAP	analyst capability
AEXP	application experience
PCAP	programmer capability
VEXP	virtual machine experience
LEXP	language experience

Table 2: Software Metrics

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
CBO	Coupling Between Object Classes

6.3 Accuracy and Prediction

The accuracy is defined as the mean magnitude of relative error (MMRE), which is the mean of percentage errors as in equation (7) [19]

$$MMRE = \sum_{i=1}^{i=n} \left(\frac{\left| E - \hat{E} \right|}{E} \right)_i \frac{100}{n} \quad , \qquad (7)$$

Table 3: Metrics information gain

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

where n is the number of projects (i.e. cases), E is the actual effort, and Ê is the predicted effort. If the value of MMRE is a large positive value, then the model over-estimates the cost, while a large negative would indicate that the model under-estimates the software cost. The MMRE is not always appropriate indicator of the prediction, where extreme deviations from the mean can affect the final prediction, so the percentage of predictions that fall within 25 percent of the actual value Pred25 (eq. 8) has been calculated for comparison, so both MMRE and Pred25 have been studied as a performance measures.

$$Pred_{25} = |P|^{-1} \left| \{ p \in P \left| |r_p| \leq 0.25 \} \right| \right. , \qquad (8)$$

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 and e_p is the actual effort [13].

6.4 Results

Authors work with three different domains (Communications, Finance, Games) of projects in the dataset used. In these experiments each domain has been used as a different data set for two reasons:

- Each of the three domains used in this study possess various software metrics. For example finance applications require larger number of code lines, game applications have a higher value of object oriented metrics, while the communications projects complexity is higher than that the financial projects. It should be noted that these aspects are based on the personal point of view and to some extent depend on the experience of the programmer.
- The efficiency is not an issue if the number of dataset cases is less than 100 cases [19].

Journal of Theoretical and Applied Information Technology
31st May 2014. Vol. 63 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The research target feature is MCDC because it has the highest information gain value which is 0.479 as shown in table 3. K in the tables above is the number of analogues. 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.

The results MMRE and Pred25 (refer to tables 4 and 5) are computed using the ANGEL analogy based tool, where both the higher Pred25 and the smaller MMRE scores mean better predictive accuracy, and have been plotted on figures 3 and 4 respectively.

Table 4: MMRE As Function Of Dataset Size And Number Of Analogies.

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

Table 5: Pred25 as function of Dataset Size and Number of Analogies.

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

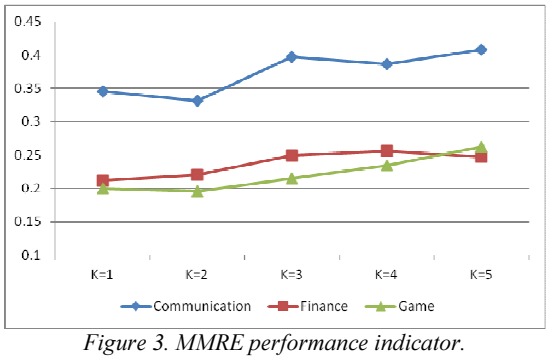

Figure 3. MMRE performance indicator.

Computing the MMRE and Pred25 needs a considerable time. For attributes represented in table 1 and table 2, about 4 hours have been needed to compute the results. Calculation time depends on the number of cases being tested. The more cases you have the more computations time you need.

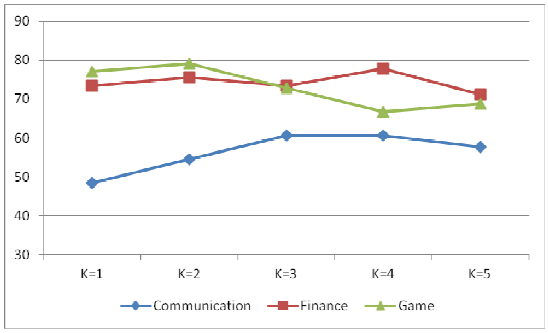

Figure 4. Pred25 performance indicator.

As for MMRE and Pred25 measures, the more cases means the more accurate measures. While Pred25 is more accurate than MMRE, both of them give 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 Pred25 measure for communication, finance, and game datasets are 56.3636, 74.222, and 72.9168 percent respectively.

6.5 Discussion

In this work, authors introduced efficient software projects cost estimation approach where the code metrics are extracted then compared to the historical ones in the database. Its main goal is to let project managers estimate the cost per releases or stages, i.e. they can extract the metrics of code implemented for a certain stage, predict the cost of the next stage of the project such giving additional knowledge to help making a proper decision in the stage agreement of certain projects. It also can be considered as a support step to further estimate the cost of development process after finishing each release. Even more, project managers can get information about risks that may pops up depending on the analysis or comparison of the code metrics of different stages. This could be one of the main documents that the project manager should pay attention before establishing a new stage.

A task of extracting features from the implemented code, estimation of the effort and deciding what can be done in the next stage can be assigned to an IT project manager. He should decide if project should be continued or abandoned depending on situation. This technique allows a

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

manager to predict the effort for a project releases or stages more accurate, if he is dealing with the same team; or provide better knowledge about the code in previous releases. Furthermore this work is the first study that applies the CBR on software projects metrics, to predict the cost per releases or stages of a certain project.

7. CONCLUSION

Predicting the cost of software projects is a big challenge to the project managers. Experienced managers can depend on their expertise to make to some extent correct estimation, while others depend on the algorithmic based models like COCOMO. Most research papers that apply the non-algorithmic approach such as 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 historical cases. In this paper authors used the software metrics of software projects to predict more precise solution to the current problem. Results show interesting performance indicators for different number of analogues. This work provided decision making support to the project manager decision in order to choose the most appropriate case to his current problem depending on the code metrics.

REFRENCES:

[1] Putnam, L.H. "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] Boehm, B., Clark, B., Horowitz, E., Madachy, R., Shelby, R., and Westland, C. "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.

[3] Myrtveit and Srensrud E. "A controlled experiment to assess the benefits of estimating with analogy and regression models". IEEE Trans. on Software Engineering, 1999, 25(4): 510 – 525.

[4] Mendes, E. and Kitchenham, B. "Further Comparison of Cross-company and Within-company Effort Estimation Models for Web Applications". Proc. of the 10th Int'l. Symp. on Software Metrics (METRICS'04), 2004, pp: 348-357.

[5] Shepperd, M., Schofield, C. and Kitchenham, B. "Effort estimation using analogy." Proceedings of the 18th international conference on Software engineering. IEEE Computer Society, 1996.

[6] Kirsopp, C., Shepperd, M., Hart, J. "Search Heuristics, Case-Based Reasoning and Software Project Effort Prediction," Empirical Software Engineering Research Group School of Design, Engineering and Computing Bournemouth University, 2002.

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

[8] Vinaykumar, K., Ravi, V., Carr, M. and Rajkiran, N. "Software cost estimation using wavelet neural networks," Journal of Systems and Software, 2008, pp. 1853-1867.

[9] Lefley, M. and Shepperd, M. "Using Genetic Programming to Improve Software Effort Estimation Based on General Data Sets", LNCS, Genetic and Evolutionary Computation — GECCO 2003, ISBN: 978-3-540-40603-7, page-208.

[10] Kumari, S. and Pushkar, S. "Comparison and Analysis of Different Software Cost Estimation Methods", International Journal of Advanced Computer Science and Applications, Vol. 4, No.1, 2013.

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

[12] Shepperd, M. and Schofield, C. "Estimating Software Project Effort Using Analogies", IEEE Transactions On Software Engineering, Vol. 23, No. 12, Nov. 1997.

[13] Najadat, H., Alsmadi, I. and Shboul, Y. "Predicting Software Projects Cost Estimation Based on Mining Historical Data", International Scholarly Research Network, ISRN Software Engineering, Vol. 2012, Article ID 823437, 8 pages.

[14] Kellner, M., Madachy, R. and Raffo, D. "Software Process Modeling and Simulation: Why, What, How." Journal of Systems and Software, 1999.

[15] Leung, H. and Fan, Z. "Software Cost Estimation, H Leung, Z Fan", Handbook of Software Engineering, Hong Kong, 2002.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[16] Jørgensen, M. and Grimstad, S. "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.

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

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

[19] Papatheocharous, E., Papadopoulos, H. and Andreou, A. "Feature Subset Selection for Software Cost Modeling and Estimation." arXiv preprint arXiv:1210.1161 (2012).

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

[21] Al-Sakran, H. "Software Cost Estimation Model Based on Integration of Multi-agent and Case-Based Reasoning," in Journal of Computer Science 2 (3): 276-282, 2006.

[22] Abu Tair, H. "Predicting The Cost Estimation Of Software Projects Using Case-Based Reasoning, ICIT 2013 The 6th International Conference on Information Technology May 8, 2013

[23] Collin Dictionary.

[24] Regolin, E., de Souza, G., etc,"Exploring Machine Learning Techniques for Software Size Estimation". Proc. of the XXIII Int'l Conf. of the Chilean Computer Science Society, IEEE, pp: 130 – 136, 2003.

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

[26] Walkerden, F. and Jeffery, R."An empirical study of analogy-based software effort estimation." Empirical Software Engineering 4.2, 1999: 135-158.

[27] Sharma, N. and Litoriya, R. "Incorporating Data Mining Techniques on Software Cost Estimation: Validation and Improvement," International Journal of Emerging Technology and Advanced Engineering, 2012, vol. 2.

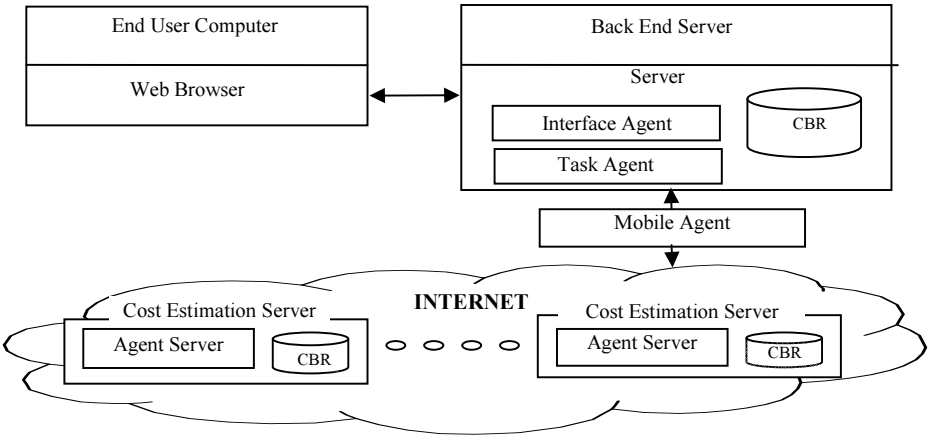

Figure 2: Architecture Of The Software Cost Estimation Model