Implementing Effort Estimation Tool as a Cloud Enabled Service

E. Sudheer Kumar
Assistant Professor, Dept of CSE,
Sree Vidyanikethan Engineering College,
A.Rangampeta, Tirupati, A.P.

V. Jyothsna
Assistant Professor, Dept of IT,
Sree Vidyanikethan Engineering College,
A.Rangampeta, Tirupati, A.P.

ABSTRACT
Software effort estimation is one of the most critical and complex, but an inevitable activity that takes place during the early stages of SDLC. Software size estimate is one of the most popular inputs for software effort prediction models. Providing a good size estimate for the purpose of accurately estimating the development effort is a challenging problem. During estimation activities, the uncertainty has become a part in software engineering measurements. The implementation of size proxy for effort estimation, which is associated with uncertainty, is a challenging task. In earlier study, there is a conceptual framework for developing size proxy, which addresses uncertainty by providing estimate as a probability density function instead of certain value. Even-though there are many estimation tools and size metrics, but none of the tool is provided as a service in cloud to addresses uncertainty issue satisfactorily. Here proposed a tool based approach by considering more predictors from various artifacts which can addresses the uncertainty issues and also providing tool as a service in cloud for users via a web browser. The tool provides output as a probabilistic value instead of certain value by considering more predictors and the results were encouraging. The tool is hosted by a vendor or service provider in the cloud and made available to customers over a network (typically the internet) having benefits like high adoption, lower initial costs, painless upgrades and seamless integration.

Keywords
Software Effort Estimation, Probabilistic Size Proxy, Pearson Correlation, Multiple Linear regression, Probability Density Function, Cloud Computing.

1. INTRODUCTION
Software effort prediction are the basis for project bidding, budgeting and planning that takes place during the early stages of development life cycle. Delivering the software on time and within budget is a critical concern for many software organizations. Underestimating software cost can have unfavorable effects on the quality of the delivered software and may affect the company's business reputation and competitiveness. Overestimation of software cost is detrimental too. It can result in missed opportunities to fund in other projects and loss of project tenders [1].

Summarizing several classes of software cost estimation models and techniques as shown in Fig.1 parametric models, expertise-based techniques, learning-oriented techniques, dynamics-based models, regression-based models, and composite-Bayesian techniques for integrating expertise-based and regression-based models. Experience to date indicates that neural-net and dynamics-based techniques are less mature than the other classes of techniques, but that all classes of techniques are challenged by the rapid pace of change in software technology. The primary conclusion is that no single technique is best for all situations, and that a careful comparison of the results of several approaches is most likely to produce realistic estimates.

Software engineering cost (and schedule) models and estimation techniques are used for a number of purposes. These include: Budgeting, Tradeoff and risk analysis, Project planning and control, Software improvement investment analysis.

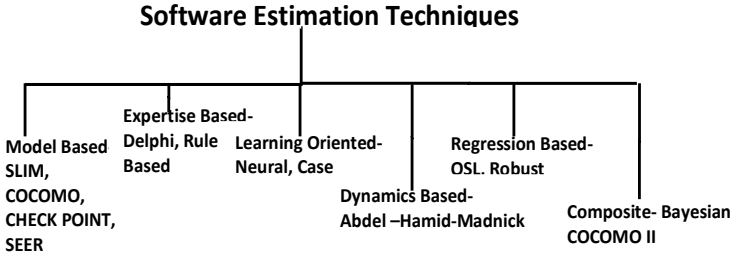


Fig 1: Types of Software Estimation Techniques

In the rest of the paper, we use the term ''size proxy'' for effort prediction rather than the term ''size metric'' to emphasize that size is measured in a way merely to predict effort and not something else, e.g., understandability. It is essential to note here that the term ''proxy'' is used to reflect that the ''size'' is being used as a proxy for the 'effort' and not the other way around. However, size estimates at the early stages of the development are the most difficult to obtain. Such estimates are often uncertain and the least accurate because very little detail is known about the project and the product at the beginning.

The uncertainty arises due to the inability to consider all factors (henceforth we will use the term predictors instead of factors to convey the idea of them taking part in effort prediction) that would contribute to the size and effort of software; i.e., neglecting some predictors due to the lack of a complete theory on software size and effort, the impracticality to use all the known predictors that contribute to software size and effort, the uncertainty in the measurement itself, etc. Thus, there are many sources of uncertainty.

Predicting the effort solely based on the number of use cases and the number of classes is expected to possess some error. The error value would reflect the amount of contribution of the neglected predictors. Accordingly, the error is not certain but rather probabilistic with non-zero standard deviation. Similarly, the information on use cases or classes could itself be uncertain, for example, the flow of events may not be accurate or subject to change. More-over, it is also possible that some use cases were overlooked or some extra use cases were added to the models. Furthermore, the way use cases are described may differ from one modeler to another. Clearly, it is not possible to just ignore these uncertainties.

The above discussion motivated our research towards achieving early estimate of effort which can be as accurate as possible and account for uncertainty at the same time. In this paper, we present a tool based approach for the development of probabilistic size proxies for effort prediction; such proxies account for uncertainty and allow for better understanding and insight into estimates. The

idea is to consider a random error, e, that is assumed to be distributed with E (e) = 0 and Var (e) = r^2 as part of the prediction model.

Proxies developed using the proposed framework predict the development effort as a probability density function (pdf) rather than a certain value. They have advantages compared to traditional metrics in that none of the existing metrics consider the random error and corresponding residual variance within their measurements.

A major difference here is that during the training step of the framework, training takes place on two different parts: mean effort and error standard deviation as explained later in the paper. Other approaches train for the mean effort only. It is important to note here that the proposed approach is different from other approaches when it comes to handling the error in estimation (a notable example is to use metric like Confidence Interval (CI), e.g. CI at 95%).

In a value added we are proposing a tool based approach which considers more predictors from various artifacts which can addresses the uncertainty issues and also providing tool as a service in cloud for users via a web browser. This is very important keeping in mind that different organizations may opt to develop different types of conceptual models early in the development lifecycle. The level of conceptual modeling done is based on many factors like the complexity of the problem, the do-main of the problem, the expertise of the modeling team.

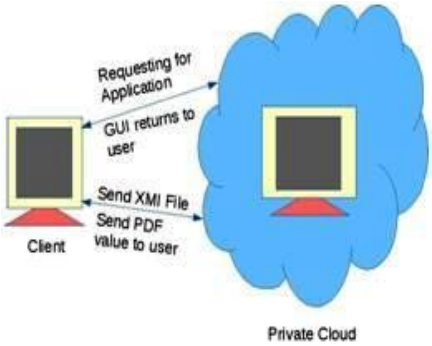


Fig 2: System Overview

Client has to request for Effort Estimation Tool in private cloud, then cloud will check all credentials of user and then it allows user to access the tool. Now user has to send his XMI file through the GUI for tool. The proposed tool process as follow: extraction of metrics from received artifacts with the help of SDMetrics tool, selecting suitable predictors, training the functions of mean effort and standard deviation error using linear regression and generating equation with selected predictors. The EET provides output as a probabilistic value instead of certain value by considering more predictors for user.

The EET is hosted by a vendor or service provider in the cloud and made available to customers over a network (typically the internet) having benefits like high adoption, lower initial costs, painless upgrades and seamless integration. We have conducted some experiments to develop a size proxy for effort prediction using our architecture. As a case study, we considered information from UML conceptual models available during the prediction activity. A major reason for using UML models is that UML is so popular and widely accepted in software industry. With object-oriented development becoming the de-facto in industry, UML's popularity has reached even new heights. Thus we believe that approaches based on UML will have high acceptability in the industry.

The rest of the paper is organized as follows. Chapter 2 presents the related work on software effort estimation, size metrics and existing system. Chapter 3 presents system architecture for the development of software size proxies for effort prediction. Chapter 4 presents experiment design, validation, results. Finally in Chapter 5 we conclude by mentioning the contributions, limitations and some possible future work.

2. RELATED WORK

This chapter presents our literature survey on software effort estimation techniques. Boehm et al. [2] provided a survey of software cost estimation techniques. They classified the software estimation techniques into six categories i.e. model based, expertise based, learning oriented, dynamics based, regression based and composite techniques. Their survey was more focused on the first category i.e. model based techniques. Briand and Wieczorek [3] provided a comparison study on different software cost/effort estimation techniques by classifying the techniques into two broad categories, "model based methods" and "non model based methods". As with the case of Boehm et al, their work was also mainly focused on model based techniques. Saliu and Ahmed [4] provided a survey on software effort estimation focusing primarily on soft computing based systems. Another good survey on effort estimation is provided by Pfleeger et el. [5]. We believe that our literature survey is an attempt to provide a more comprehensive study on effort estimation rather than just looking at individual works.

Below we discuss the major software effort estimation techniques by broadly dividing them into four categories namely, "expert judgment", "analogy", "algorithmic models" and "models based on soft computing". This classification is based on our own survey of different software effort estimating techniques and to follow our evaluation of these techniques. It should be noted that there is no agreed classification of software effort estimation techniques and the classifications are subjective [6] [7] [8] [9] [10] [11] [12].

In this section we discuss the prominent software size metrics found in the literature. First of all we discuss the metric Lines of Code (LOC) followed by Function Points (FP) and its various object oriented and non-object oriented variations. This is followed by discussion on Use Case Points (UCP), Class Points (CP), Vector Size Measure (VSM), Number Of Components (NOC), Object Point, Fast&&Serious and Predictive Object Point (POP) [13] [14] [15].

2.1 Existing System

It is worth noting here discusses a conceptual framework that outlines a possible approach to tackle the issue but does not dictate specific settings of the parameters involved (e.g., the predictors to consider); rather, the parameters are typically set to suit the data available

Presenting a conceptual framework for the development of size proxies, such size proxies inherently portray the probability distribution of the error associated with their corresponding computed effort estimates. Such a portrayal allows associating a degree of confidence to the effort estimate for a given project.

It is important to note, though, just present here a conceptual framework which outlines a possible approach to tackle the issue but does not dictate specific settings of the parameters involved (e.g., the predictors to consider); rather, the parameters are typically set to suit the data available. Given available historical data, practitioners can use the framework to develop a suitable size proxy for effort prediction. Clearly, the richer the historical data, the higher the confidence on the resultant size proxy.

To account for the uncertainty, the framework can be used to generate size proxies, each of which is composed of two components: an estimate and an error (the term 'residual' rather than error is used commonly is statistics).

The estimate represents the mean effort measurement computed using the predictors considered; whereas the error (i.e., residual

variance) represents the standard deviation that accounts for the uncertainty in prediction.

It is important to mention here that the higher the standard deviation of estimate, the lower the confidence in the estimate; i.e., the higher the uncertainty in the computed estimate.

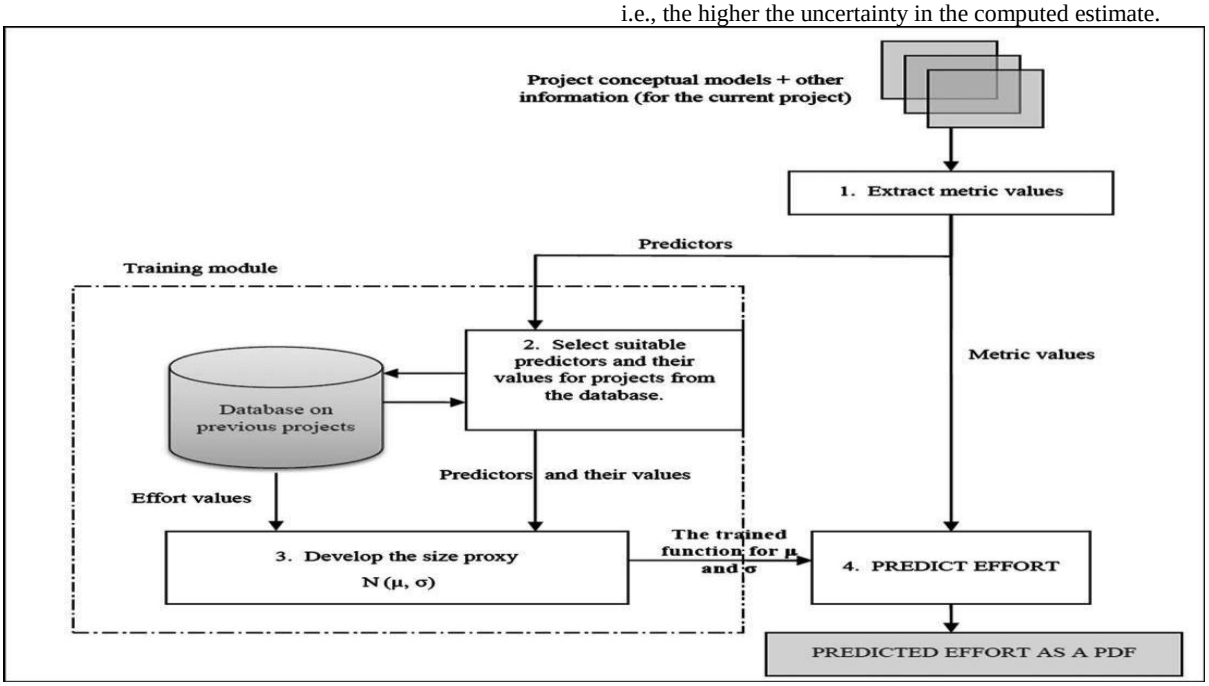


Fig 3: A Conceptual Framework for Probabilistic Size Proxy

3. ARCHITECTURE DESIGN

The main purpose of this design is to develop an Effort Estimation Tool which generates a probabilistic size proxy to predict effort required to develop a project in the early stages of software development life cycle. This tool can be used in real time scenarios for prediction activities, when they have conceptual diagrams for a future projects. The tool architecture can be used to convert artifact in to metrics extraction, training the size proxy, predict effort and produce a probability density function value. With this architecture the uncertainty issue will be solved satisfactorily.

The extraction process inside the architecture can be done by using SDMetrics tool, selecting the suitable predictors can be done with the help of Pearson correlation, providing training to develop the size proxy by using linear regression technique and also provide results for different confidence intervals.

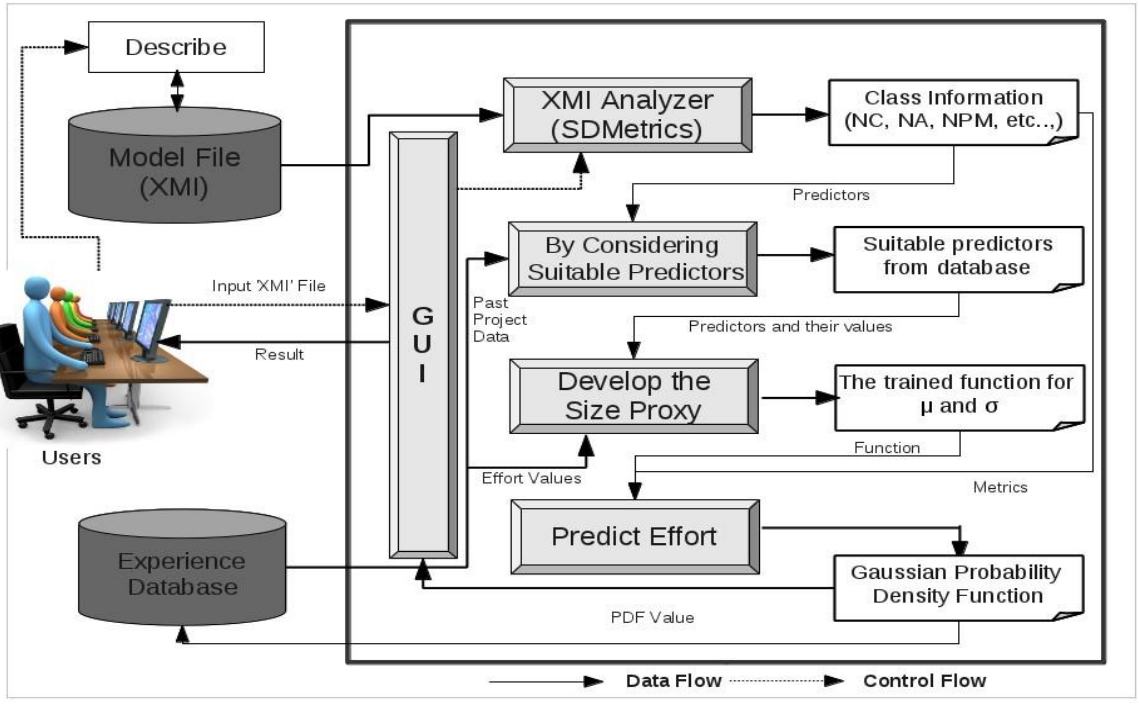


Fig 4: Proposed Architecture for Probabilistic Size Proxy

Step1: Metric value extraction: This component extracts the metric values from the current project for which the effort needs to be estimated. This component takes Project's conceptual models i.e. the UML artifacts along with any other project specific information like the technical factors affecting the project.
Step2: Selection of metric and their values from past projects: This component selects the size proxy constituents i.e. the predictors for effort, based on the metrics whose values are available for the current project whose effort needs to be estimated and also based on the metrics for which we have information available for sufficient number of past projects in the database.
Step3: Training of the size proxy: After selecting the predictors for effort estimation, the predictors along with their values are provided to the training component. The effort values for past completed projects are also provided to the training component from the database. The training component can use one of the different training techniques like regression, neural networks, genetic programming to train the function for mean (μ) and the standard deviation (σ) of the size proxy.
Step4: Effort estimation: The trained functions for mean (μ) and standard deviation (σ) are provided to the effort estimation component. This component obtains the metric values for the current project from the component one and uses them in the functions for mean (μ) and standard deviation (σ) to provide the effort estimate as a PDF.

3.1 Metrics Table Data

Whenever user input xmi file through gui, the file will forwarded to SDMetrics tool. The tool analyzes the file; calculate metrics and then the results will be generated in the specified format as shown in below Table 3.1.
The output file from SDMetrics tool has individual classes of a project and its metrics values. Now we have to calculate overall values for each and every predictor and store it in to another table as shown in below Table 3.2.
Past projects data has been generated using simulation method. Each and every dependent and independent variables value has been generated and with the help of this data estimated μ can be calculated by using linear regression technique. In general, multiple regression procedures will estimate a linear equation of the form as shown in equation (1):

$$Y = a + b_1*X_1 + b_2*X_2 + \ldots\ldots + b_p*X_p \qquad (1)$$

Table: 3.1 SDMetrics Output File Format

Name	Num Attr	Num Ops	NumPub Ops	……	MsgSelf
Class 1	0	1	2		6
Class 2	1	2	3		3
Class n	2	3	5	……	4

3.2 Estimated Mean Data

Calculating mean equation with the help of linear regression, and the dataset structure is tabulated in Table 3-2 and formula is:

$$\text{Mean Estimated Effort } (\mu_{effort}) = b_0 + \sum_{i=1}^{m}(bi \times Ai) \qquad (2)$$

Where,

'm' = number of predictors (independent variables)
' b_i ' = Beta coefficient for variable 'i'.
' A_i ' = Independent/predictor variable 'i'

Table: 3.2 Input dataset structure for training mean effort

P. No	Dependent Variable (Effort)	Independent Variable (A1)	IV (A2)	……	IV (Am)
1	240	20	18		19
2	350	30	14		15
		……			
N	…..	…….	…..	……	…….

3.3 Standard Deviation Data

Each perturbated set/cluster number consists of number of projects as one set and also with their corresponding values. The data represented here has different independent variables and standard deviation of error estimate variable which is used for calculating the linear regression equation of standard deviation error along with mean error as of like in formulae (3).
Mean error formula

$$\mu^{i}_{error} = \frac{1}{N}\sum_{j=1}^{N}(E^{i}_{j} - \mu^{i}_{effort,j}) \qquad (3)$$

Where,

μ (i) = mean error estimate for perturbed set 'i'.
N = Number of projects in each perturbed set.
E ij = Effort for perturbed project 'j' in perturbed set 'i' effort
μi = Estimated effort for project 'j' in perturbed set i.
σ (i) = Standard deviation of errors (μ) for the 'N' projects in perturbed set 'i'.

3.4 Standard Deviation of Error Estimate

$$\sigma(i) = \sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(\mu^{i}_{error} - \left(E^{i}_{j} - \mu^{i}_{effort\ j}\right)\right)^{2}} \qquad (4)$$

Where

r (i) = standard deviation of error in estimation for perturbation set 'i'

N = total number of projects in each perturbed set

μ_{effort} = mean error in estimation for perturbation set 'i'

E_k = $effort_j$ for perturbed project 'j' in perturbed set 'i'

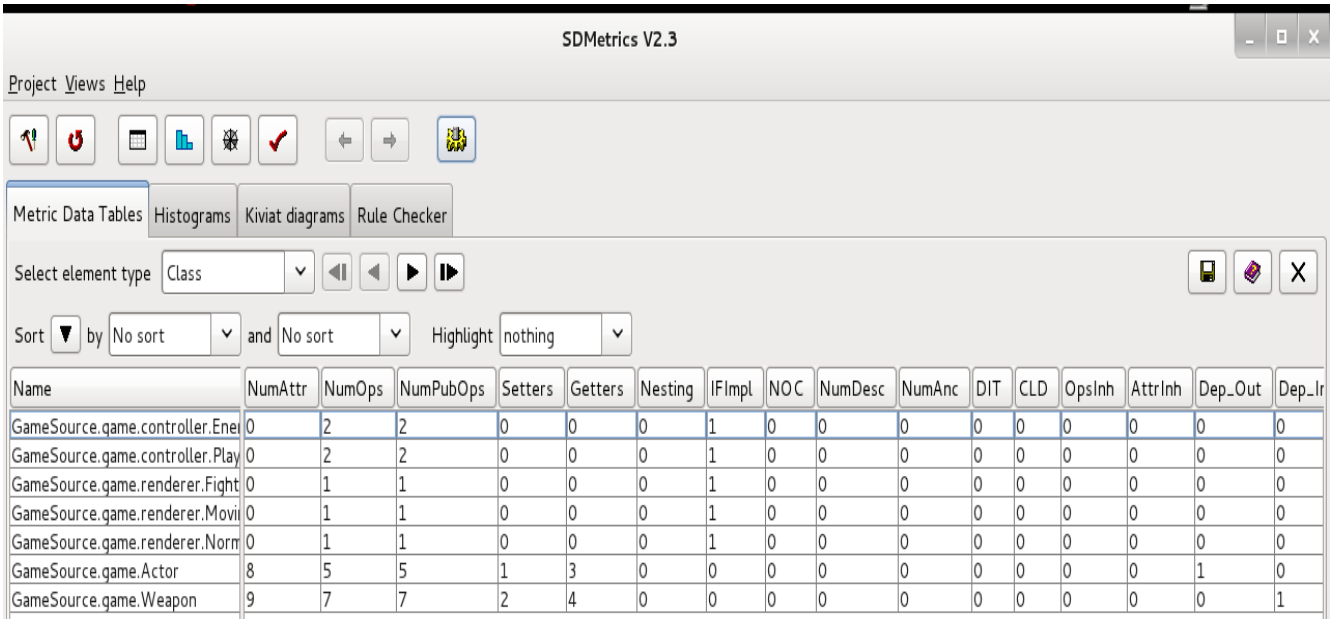
μ_{effort} = estimated mean effort for project 'j' in perturbed set i.

3.5 Cloud Installation and Application Hosting

The product features automated deployment of cloud stack and provided a convenient environment through portal for infrastructure request, application hosting, view usage and billing information and administrator users to manage their entire cloud environment and security administrators to monitor the security violations.
Insert the Meghdoot DVD into the drive, restart your computer and boot from DVD by editing the BIOS setup. The screen appears with three options:

- Start Meghdoot (BOSS CLOUD) Live
- Install Meghdoot (BOSS CLOUD)-Graphical
- Install Meghdoot (BOSS CLOUD)-Text Mode

You can proceed with the default installation by clicking "Install Meghdoot (BOSS CLOUD) - Graphical" or "Install Meghdoot (BOSS CLOUD)-Text Mode".



Name	NumAttr	NumOps	NumPubOps	Setters	Getters	Nesting	IFImpl	NOC	NumDesc	NumAnc	DIT	CLD	OpsInh	AttrInh	Dep_Out	Dep_In
GameSource.game.controller.Ener	0	2	2	0	0	0	1	0	0	0	0	0	0	0	0	0
GameSource.game.controller.Play	0	2	2	0	0	0	1	0	0	0	0	0	0	0	0	0
GameSource.game.renderer.Fight	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0
GameSource.game.renderer.Movi	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0
GameSource.game.renderer.Norm	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0
GameSource.game.Actor	8	5	5	1	3	0	0	0	0	0	0	0	0	0	1	0
GameSource.game.Weapon	9	7	7	2	4	0	0	0	0	0	0	0	0	0	0	1

Fig 5: Output of SDMetrics Tool

4. EXPERIMENTAL SETUP

4.1 SDMetrics Tool

The Fig 4.1 is the first and foremost step which is going to happen with the help of user. The user will connect to tool through web graphical user interface and give XMI file of a class diagram for a project. The SDMetrics tool will extract the metrics form that class diagram and stored in the form of table format as shown above fig 4.1.

4.2 Correlation Table

Pearson Correlation Coefficient is used, inorder to find out (linear) relationship between the two variables x and y. Based on the past data available in database, it retrieves data from database and finding pearson correlation between effort variable and all remaining variables. Table 4.1 shows correlation table:

Table: 4.1 Correlation table based on Past Data

	NC	NA	NPM	IFImpl	Num Desc	Num Anc	DIT
Effort (Man-Hours)	0.98	0.99	0.99	0.98	0.28	0.97	0.98

4.3 Size Metric

The size metric is one of the main inputs used to find effort estimation. There it needs training to form a metric, so we are considering multiple linear regression as our training technique. It is a combination of mean, error and standard deviation form. Mean will calculate based on the past data, where as it will consider effort variable independent variable and all other remaining variables as dependent variables, so that we will get an intercept value and coefficients value to form a equation. Standard deviation will also calculate based on the past data, where as it will consider standard deviation error.

First of all we need to calculate error value by considering difference between actual effort values and estimated mean effort values of past projects. We need to perform standard deviation error with the help of error value for all past projects. Consider standard deviation error value as independent variable and remaining variables as dependent variable to get intercept value and coefficients values. With the help of mean and standard deviation error values, the size proxy equation is formed.

$$E_{estimated} = N\,[(311.39 + 05.84 * NC + 03.17 * NA + 05.04 * NPM + 00.15 * IFImpl + 00.70 * NumDesc + -01.02 * NumAnc + 00.52 * DIT), (-13.08 + 02.59 * NC + -00.53 * NA + 00.46 * NPM + -00.07 * IFImpl + 00.71 * NumDesc + 00.49 * NumAnc + -00.48 * DIT)]$$

4.4 PDF Value

Here we need to substitute the variables value from the user table database and it shows the values of mean and standard deviation error. Now we need to calculate Gaussian probability density function value by adding +/- of 1.96*SDE value to mean, so that we can get probabilistic value instead of certain value

The Gaussian Probability Density Function Value is:

$E_{estimated}$ = N [**403.35, 10.99**]

Effort Estimate for the confidence interval of **95%** is:

E [**381.80, 424.90**]

5. CONCLUSION AND FUTURE WORK

A tool based approach for effort estimation was proposed, which can be used early in software development process. It is flexible to the amount of information available during estimation and does account for uncertainty by providing estimate as a Gaussian PDF. The Architecture was validated by creating instantiating size proxies consisting of different number of predictors and validating these size proxies using two different data sets. User has to request for Effort Estimation Tool in private cloud, then cloud will check all credentials of user and then it allows user to access the tool. Now user has to send his XMI file through the GUI to tool. The tool is hosted by a vendor or service provider in the cloud and made available to customers over a network (typically the internet) having benefits like high adoption, lower initial costs, painless upgrades and seamless integration, so that user can access from anywhere with the help of browser and does not require any additional resources.

5.1 Limitations and Future Work

Following are some of the major limitations of our work along with proposed future works:

- A major limitation of our work is the shortage of data. Shortage of data is a typical problem faced by research community in software engineering. Dataset one had data on 20 projects with simulation / trial and error. Thus validating the proxy using more data and preferable from industry projects is a definite future work.
- The architecture may also be extended by adding steps to handle imprecision especially when using attributes having categorical values or inputs from expert. Fuzzy logic is a good choice to model imprecision in such cases.
- Another possible future work is to use the intermediate size proxy by providing the non-singleton size measure to effort estimation models such as the one proposed by zeeshan.

We used regression analysis to train our size proxy. Other training algorithms and techniques like ANN can also be used and the results can be compared.

6. REFERENCES

[1] Moataz A. Ahmed, Irfan Ahmad, Jarallah S. AlGhamdi "*Probabilistic size proxy for software effort prediction: A framework*" Information and Software Technology, Volume 55, Issue 2, February 2013, Pages 241-251

[2] Boehm, Barry, Abts, Chris and Chulani, Sunita, J.C. Baltzer "*Software development cost estimation approaches – A survey*", Annals of Software Engineering 10 (2000), AG, Science Publishers. Page(s): 177-205.

[3] Briand, Lionel C. and Wieczorek, Isabella "*Resource Estimation in Software Engineering*", Technical Report, International Software Engineering Research Network.

[4] Saliu, M.O. and Ahmed, M.A., A Chapter in E. Damiani, L. C. Jain, and M. Madravio (EDs), "*Soft Computing Based Effort Prediction Systems – A Survey*", Soft Computing in Software Engineering, Springer-Verlag Publisher, July 2004, ISBN 3-540-22030-5.

[5] Pfleeger, Shari Lawrence Wu' Felicia and Lewis Rosalind, *Software Cost Estimation and Sizing Methods, Issues and Guidelines*, RAND project Air Force 2005.

[6] Briand, Lionel C. and Wieczorek, Isabella "*Resource Estimation in Software Engineering*", Technical Report, International Software Engineering Research Network.

[7] Anda, Bente "*Comparing Effort Estimates Based on Use Case Points with Expert Estimates*", EASE 2002-Empirical Assessment in Software Engineering, Keele, UK, April 8-10,2002.

[8] Mendes, Emilia, Mosley, Nile Counsell, Steve, "*Do Adaptation Rules Improve Web Cost Estimation?*" Proceedings of the fourteenth ACM conference on Hypertext and hypermedia, August 2003.

[9] Cuadrado-Gallego, Juan J. Sicilia, Miguel-A´ngel Garre, Miguel and Rodrı´guez, Daniel; "*An empirical study of process-related attributes in segmented software cost-estimation relationships*", Journal of Systems and Software, Volume 79, Issue 3, March 2006, Pages 353-361.

[10] Russell, Stuart and Norwig, Peter, Artificial Intelligence, a modern approach, second edition, prentice hall, 2003.

[11] Tadayon, N. "*Neural Network Approach for Software Cost Estimation*", Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05), Volume 2, 4-6 April 2005, Page(s):815 – 818.

[12] Danny, Xishi Huang Ho, Luiz, Jing Ren, Capretz, F. "*A soft computing framework for software effort estimation*", Soft Computing (2006) 10: 170–177, Springer.

[13] Tan, Hee Beng Kuan, Zhao, Yuan and Zhang, Hongyu "*Estimating LOC for Information Systems from their Conceptual Data Models*", proceeding of the 28th international conference on Software engineering, Pages: 321 – 330.

[14] Albrecht, A.J. and J.R. Gaffney "*Software function, source lines of code, and development effort prediction a software science validation*", IEEE Transactions on Software Engineering, Volume SE-9, Issue 6, Nov. 1983 Page(s):639 – 648.

[15] Costagliola, G. Ferrucci, F. Tortora, G. Vitiello, "*Class point: an approach for the size estimation of object-oriented systems*", IEEE Transactions on Software Engineering, Volume 31, Issue 1, Jan. 2005 Page(s):52 – 74.