Abstract—Effective estimation is very necessary for efficient design of an object-oriented software. Software developers need to read and understand source programs and other software artifacts. The increase in size and complexity of software considerably affects a number of quality attributes, especially effectiveness and testability. False analysis frequently leads to ambiguities, misunderstanding and hence to faulty development results. It is very tough to obtain an understandable view on all the probable factors that have positive impact on software testability. Researchers, practitioners and quality controllers have always argued that effectiveness should be measured as a key attribute in order to promise the quality software. Calculating effectiveness at source code level directs to late arrival of desired information. An exact measure of software quality fully depends on effectiveness estimation. This paper shows the results of a systematic literature review conducted to collect related evidence on effectiveness estimation of object-oriented software. In this paper, our objective is to find the known complete and comprehensive software effectiveness estimation model and related framework for estimating the effectiveness of object-oriented software at an initial stage of development life cycle.

Keywords: Software Effectiveness; Quantification; Object Oriented Design Characteristics; Software Quality; Software testing, Aspect Oriented Design Characteristics.

1. INTRODUCTION

There are numerous approaches to make the system very much reliable. Among several available methods object oriented design is one of the important methods to measure effectiveness and design effectiveness [1, 2]. Aspect oriented design signed itself as an essential approach for resolving mainly of the software problems. In an object oriented approach, the data is considered as the most significant element and it cannot move freely around the system [16]. Increase in the size of program, increases needless effort and complexity. Software complexity always increases with error handling functions. Software with high complexity generally produces software with faults [5]. High complexity always decreases effectiveness of software. Though, software faults vary noticeably with respect to their severity. A failure occurred by a fault may go ahead to a complete system crash or an incapability to open a file [12, 13]. Therefore, there is an urgent need to develop a model that can be applied to identify those classes that are prone to have serious faults. From the abovementioned discussion, it appears that reducing unwanted complexity early in the development process leads to the development of high quality reliable end products. Effectiveness is an essential software quality factor that is useless if it is not available at an initial stage in the software development life cycle. It becomes more important in the case of object-oriented design. This chapter illustrates the need and significance of effectiveness at design phase and build up a multivariate linear Effectiveness.

II. SOFTWARE EFFECTIVENESS

Effectiveness has always been an indefinable concept. Its truthful measurement or assessment is a complex exercise for the reason that of the various potential factors influencing effectiveness. It has been found out from systematic literature review that area researchers, quality controllers and industry personnel had made significant efforts to estimate software effectiveness but at the source code level. Calculating effectiveness at source code level directs to late arrival of desired information. An exact measure of software quality fully depends on effectiveness estimation. This paper shows the results of a systematic literature review conducted to collect related evidence on effectiveness estimation of object-oriented software. [3, 4, 7]. The majority of companies pay out over 70 percent budget on testing, maintenance of the software to manage the quality [5]. Effectiveness Estimation helps to examine the maintenance effort and easiness of software at design level [14]. The effectiveness definition according to IEEE glossary of Software Engineering is “the ease with which a software system or component can be modified to correct faults, get better performance or other attributes, or adapt to a change environment” [6, 23].

Software identification processes normally focus on avoiding errors, detecting and correcting software faults that do occur, and predicting effectiveness after identification. It is well understood that delivering quality software is no longer an advantage but a necessary factor. Unluckily, the majority of the industries not only fails to deliver a quality product to their customers, but also do not understand the Estimation Model for Object-Oriented Design. Developed model estimates the effectiveness of class diagrams in respect of their Effectiveness, Effectiveness. Lastly the developed models have been validated using experimental tryout. Appropriate quality attributes [20]. Software maintenance required for extra effort than any other software engineering activity [5]. The effectiveness of software is not possible directly, but with
the help of their internal characteristics Estimation [6]. Noticeably a definition of effectiveness wishes to be strongly linked to the term maintenance. Effectiveness is the easiness or simplicity with which a software system can be maintained (using the definition of software maintenance above) and is a key characteristic of software [16, 18].

III. EFFECTIVENESS MANAGEMENT

Key aspects of effectiveness management include:

- Corporate level participation
- Integral piece of product identification not parallel
- Effectiveness procedures incorporated into design process
- Built into programme plan and create an effectiveness plan
- Ownership of the effectiveness plan inside the design team

IV. OBJECT AND ASPECT ORIENTED DESIGN CONSTRUCTS

Object oriented design and development are well-known conceptions in today’s software development environment. Object oriented design supports number of design properties such as coupling, cohesion, inheritance and encapsulation. Object oriented system consider object as the primary agent involved in a computation process. It requires more significant effort at the early phase in the software development life cycle to recognize objects, classes, and the relationships among them. Object oriented programming is a basic knowledge that supports quality objectives [13, 15]. The necessity to deal with the effectiveness of software design is the essential issue that influenced the overall development cost and quality. A good object oriented design needs design procedures and practices that must be used in development cycle [17]. Their violation will ultimately have a strong impact on the quality attributes. Object oriented principles direct the designers what to hold up and what to keep away from. A number of measures have been defined so far to measure object oriented design. There are several important themes of object orientation that are known to be the basis of internal quality of object oriented design and support in the perspective of estimation [18,19]. These themes significantly include inheritance, encapsulation, cohesion and coupling.

VI. OBJECT AND ASPECT ORIENTED DESIGN METRICS

The most central aim of metric selection is to pick such metrics which are statistically important and must be applicable. Studies have been conducted and found that there exists powerful relation among Object Oriented software metrics and its effectiveness. Software metrics offer an effortless and inexpensive way to identify and correct probable reasons for low software quality according to the effectiveness sub-factor as this will be supposed by the programmers. Set up Estimation programs and design metric standards will support in preventing failures before the maintenance process and decrease the essential effort during that phase. Internal metrics are extremely associated with the programmers’ view of effectiveness [9-12]. However, unhappiness with internal quality standards may not necessarily outcome in low rank of effectiveness although it is generally expected. In that case, it is likeable that, regardless of what internal Estimations designate, the concluding judge for the effectiveness of the delivered software is the programmer [19, 21, 22].

VII. MODELS DEVELOPMENT

Estimation of class diagram’s Effectiveness is a prerequisite for the accurate effectiveness Estimation. For this reason prior to developing EEMOOD, the study has developed models for Effectiveness. In order to set up the models subsequent multivariate linear model (1) has selected.

\[ Y = \mu + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \varepsilon \] (I)

Where
- \( Y \) is dependent variables.
- \( X_1, X_2 \ldots X_n \) are independent variables.
- \( \beta_1, \beta_2 \ldots \beta_n \) are the coefficients.
- \( \varepsilon \) is error term
- \( \mu \) is the intercept.

4.5.1 Effectiveness Estimation Model
In order to set up an Effectiveness estimation model of object oriented class diagram, metrics listed in [8] will play the role of independent variables while Effectiveness will be taken as dependent variable. The data used for developing Effectiveness model is taken from [10]. SDMetric software has been used to calculate the values of independent variables. Firstly , class diagram has been converted into XMI file and using sdmetric , xmi file has been used to calculate the values of independent variables. The correlation among Effectiveness Factors and Object Oriented Characteristics has been established as depicted in equation2. Using SPSS, values of coefficient are calculated and Effectiveness model is originated as below.
### Name: Assoc

**Domain:** diagram

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumAttr</td>
<td>Class</td>
<td>Size</td>
<td>The number of attributes in the class.</td>
</tr>
<tr>
<td>NumOps</td>
<td>Class</td>
<td>Size</td>
<td>The number of operations in a class.</td>
</tr>
<tr>
<td>Num/op</td>
<td>Class</td>
<td>Size</td>
<td>The number of pubic operations in a class.</td>
</tr>
<tr>
<td>Setters</td>
<td>Class</td>
<td>Size</td>
<td>The number of operations with a name starting with 'set'.</td>
</tr>
<tr>
<td>Getters</td>
<td>Class</td>
<td>Size</td>
<td>The number of operations with a name starting with 'get', 'is', or 'has'.</td>
</tr>
<tr>
<td>nesting</td>
<td>Class</td>
<td>Inheritance</td>
<td>The nesting level of the class (for inner classes).</td>
</tr>
<tr>
<td>E1impl</td>
<td>Class</td>
<td>Inheritance</td>
<td>The number of interfaces the class implements.</td>
</tr>
<tr>
<td>NCL</td>
<td>Class</td>
<td>Inheritance</td>
<td>The number of children of the class (UML Generalization).</td>
</tr>
<tr>
<td>NumDesc</td>
<td>Class</td>
<td>Inheritance</td>
<td>The number of descendants of the class (UML Generalization).</td>
</tr>
<tr>
<td>NumAnc</td>
<td>Class</td>
<td>Inheritance</td>
<td>The number of ancestors of the class.</td>
</tr>
<tr>
<td>DDT</td>
<td>Class</td>
<td>Inheritance</td>
<td>The depth of the class in the inheritance hierarchy.</td>
</tr>
<tr>
<td>Gold</td>
<td>Class</td>
<td>Inheritance</td>
<td>The class to leaf depth.</td>
</tr>
<tr>
<td>Opsh</td>
<td>Class</td>
<td>Inheritance</td>
<td>The number of inherited operations.</td>
</tr>
<tr>
<td>Attr</td>
<td>Class</td>
<td>Inheritance</td>
<td>The number of inherited attributes.</td>
</tr>
<tr>
<td>Dept_out</td>
<td>Class</td>
<td>Coupling (import)</td>
<td>The number of elements on which this class depends.</td>
</tr>
<tr>
<td>Dept_in</td>
<td>Class</td>
<td>Coupling (export)</td>
<td>The number of elements that depend on this class.</td>
</tr>
<tr>
<td>Numelemi</td>
<td>Class</td>
<td>Coupling</td>
<td>The number of associated elements in the same scope (namespace) as the class.</td>
</tr>
<tr>
<td>Numelmb</td>
<td>Class</td>
<td>Coupling</td>
<td>The number of associated elements in the same scope branch as the class.</td>
</tr>
<tr>
<td>BC</td>
<td>Class</td>
<td>Coupling (import)</td>
<td>The number of bases the class is externally used as attribute type.</td>
</tr>
<tr>
<td>BCAttr</td>
<td>Class</td>
<td>Coupling (import)</td>
<td>The number of bases in the class having another class or interface as their type.</td>
</tr>
<tr>
<td>BCPar</td>
<td>Class</td>
<td>Coupling</td>
<td>The number of parameters in the class having another class or interface as their type.</td>
</tr>
<tr>
<td>Connectors</td>
<td>Class</td>
<td>Complexity</td>
<td>The number of connectors owned by the class.</td>
</tr>
<tr>
<td>Instspec</td>
<td>Class</td>
<td>The number of instance specification where the class is a classifier.</td>
<td></td>
</tr>
<tr>
<td>LInherit</td>
<td>Class</td>
<td>The number of instances that represent a property of which this class is the type.</td>
<td></td>
</tr>
<tr>
<td>Instspec</td>
<td>Class</td>
<td>Coupling (import)</td>
<td>The number of messages sent.</td>
</tr>
<tr>
<td>Instspec</td>
<td>Class</td>
<td>Coupling (import)</td>
<td>The number of messages received.</td>
</tr>
</tbody>
</table>
First, values from different criteria’s has been calculated for every independent variable and then avg of these values has been taken as the value of an independent variable.
Table 2. Model Summary

<table>
<thead>
<tr>
<th>Model</th>
<th>R</th>
<th>R Square</th>
<th>Adjusted R Square</th>
<th>Std. Error of the Estimate</th>
<th>Change Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>.999&lt;sup&gt;a&lt;/sup&gt;</td>
<td>.996</td>
<td>.988</td>
<td>.34864</td>
<td>.996</td>
</tr>
</tbody>
</table>

a. Predictors: (Constant), Hierarchies, Inheritance, Encapsulation, Coupling

Table 3. ANOVA<sup>b</sup>

<table>
<thead>
<tr>
<th>Model</th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Regression</td>
<td>48.645</td>
<td>4</td>
<td>12.161</td>
<td>99.540</td>
</tr>
<tr>
<td></td>
<td>Residual</td>
<td>.122</td>
<td>1</td>
<td>.122</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>48.767</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. Predictors: (Constant), Hierarchies, Inheritance, Encapsulation, Coupling
b. Dependent Variable: Effectiveness

Table 1. Coefficients<sup>a</sup>

<table>
<thead>
<tr>
<th>Model</th>
<th>Unstandardized Coefficients</th>
<th>Standardized Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
<td>Std. Error</td>
</tr>
<tr>
<td>1</td>
<td>(Constant)</td>
<td>-4.079</td>
</tr>
<tr>
<td></td>
<td>Encapsulation</td>
<td>4.599</td>
</tr>
<tr>
<td></td>
<td>Inheritance</td>
<td>12.000</td>
</tr>
<tr>
<td></td>
<td>Coupling</td>
<td>2.698</td>
</tr>
<tr>
<td></td>
<td>Hierarchies</td>
<td>-0.507</td>
</tr>
</tbody>
</table>

a. Dependent Variable: Effectiveness

Effectiveness Estimation Model Output and Result Summary

The Coefficients part of the output gives us the values that we need in order to write the regression equation (4). The Standardized Beta Coefficients give a measure of the contribution of each variable to the Effectiveness model. A big value designates that a unit change in this predictor variable has a large effect on the criterion variable. The t and Sig (p) values give a rough indication of the impact of each predictor variable – a big absolute T value and small p value suggests that a predictor variable is having a large impact on the criterion variable. The experimental evaluation of Effectiveness is very encouraging to obtain effectiveness index of software design for low cost testing and maintenance.

Table 4.1: Coefficients for Effectiveness Estimation Model

The descriptive statistics of the output gives the mean, standard deviation, and observation count (N) for each of the dependent and independent variables and is shown in Table 4.2.

Table 4.2: Descriptive Statistics for Effectiveness Estimation Model

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effectiveness</td>
<td>8.1773</td>
<td>3.1646</td>
<td>6</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>0.3816</td>
<td>0.1531</td>
<td>6</td>
</tr>
<tr>
<td>Inheritance</td>
<td>0.5416</td>
<td>0.1281</td>
<td>6</td>
</tr>
<tr>
<td>Coupling</td>
<td>1.7100</td>
<td>1.5144</td>
<td>6</td>
</tr>
<tr>
<td>Hierarchies</td>
<td>6.0000</td>
<td>11.2783</td>
<td>6</td>
</tr>
</tbody>
</table>

4.6 Empirical Validation

Empirical validation is a vital phase of proposed research. Empirical validation is the standard approach to justify the model approval. Taking view of this truth, practical validation of the effectiveness model has been performed using sample tryouts. In order to validate developed effectiveness model the data has been taken from [10].
Sperman’s Coefficient of Correlation $r_s$ was used to check the significance of correlation among calculated values of effectiveness using model and it’s ‘Known Values’. The

$$r_s = 1 - \frac{6 \sum d^2}{n(n^2 - 1)}$$

‘d’ = difference between ‘Calculated ranking’ and ‘Known ranking’ of effectiveness.

The correlation values between effectiveness through model and known ranking are shown in table (4) above. Pairs of these values with correlation values $r_s$ above $\pm 0.781$ are checked in table. The correlations are up to standard with high degree of confidence, i.e. up to 99%. Therefore we can conclude without any loss of generality that effectiveness Estimation model measures are really reliable and significant and applicable.

4.7 CONCLUSION

The study has developed model to compute effectiveness, effectiveness of the class diagrams. Effectiveness model measures the effectiveness of class diagrams in terms of their design constructs. Effectiveness model have been developed using the method of multiple linear regressions. The study moreover validates the quantifying ability of effectiveness model. The applied validation on the effectiveness model concludes that proposed model is highly consistent, acceptable and considerable.

The values of effectiveness are of instant use in the software development process. These values help software designers to review the design and take proper corrective measures, early in the development cycle, in order to control or at least reduce future maintenance/testing.

REFERENCES


Call for Papers 2018

International Journal of Engineering Research & Technology

☑ Fast, Easy, Transparent Publication
☑ More than 50000 Satisfied Authors
☑ Free Hard Copies of Certificates & Paper

Publication of Paper: Immediately after Online Peer Review

Why publish in IJERT?
☑ Broad Scope: high standards
☑ Fully Open Access: high visibility, high impact
☑ High quality: rigorous online peer review
☑ International readership
☑ Retain copyright of your article
☑ No Space constraints (any no. of pages)

Submit your Article

www.ijert.org