Chapter 7

Conclusion, Findings and Areas for Further Research

7.1 Introduction

In this chapter the scholar describes her own research results. Further the objectives are highlighted along with the research hypotheses underlined for this work and explained how these objectives are fulfilled and research hypotheses are proved or disproved. At last the research trend is described and how further research will be beneficial in this specified area are also portrayed.

7.2 Conclusion

After acquiring the background knowledge of this research work the basic objectives were set before carrying out the actual research. The primary objective of this research is to examine the existing object-oriented metrics to illustrate the need of new measures of early phase i.e. design phase, proposing four design metrics for early prediction of faults at class level, so that fault forecasting technique could be used at design phase to enhance the reliability of the final product. In all, the research work is organized to fulfill the four main objectives mentioned in chapter 1.

The first objective is ‘to examine the existing object-oriented software metrics for further use in predicating the faults in a class that affect product reliability’. To fulfill this objective the fault prediction analysis is carried out in chapter 4 by taking the set of metrics, most frequently used in the industries and researches (i.e. CK-metrics and Size metrics) for the evaluation of object-oriented system by taking into consideration the severity of faults. In this way the classes having severer faults are identified, which affects the reliability of product at most.

To accomplish the second objective — ‘to illustrate the need of new measures to identifying the faults and problem areas early in the software development process’, the results of similar type of researches i.e. for evaluating the object-oriented system using
metrics and results of Chapter 4 are analysed and it is observed that metrics for measuring the class size/complexity, cohesion and coupling are more significant in predicting the faults in comparison to other category of metrics like inheritance metrics, but most of them are evaluated using source-code. The same is also supported by the results of my study which was conducted as a pilot study to identify the metrics as design metrics from a pool of 48 existing object-oriented metrics included in chapter 2. Thus the need of new measures for early faults prediction and identification of problem areas in the software development process is justified.

To fulfill the third objective — ‘to propose a new metrics suite to find out class complexity at design level in an object oriented system’ four metrics are proposed in chapter 5 to measure the class complexity induced by the design concepts like data hiding, inheritance, aggregation and cohesion using the static information available at the end of design phase. These metrics are internally validated using two successive frameworks given by Weyuker and Braind et al. for software metrics and complexity metrics respectively.

The final objective of this study is — ‘to validate the proposed design metrics as early predictor of fault contents to enhance reliability.’ This objective is accomplished by externally validating the newly proposed metrics in chapter 6. These are theoretically and empirically validated to predict the faults in a class at design phase. The faults predicted are used to identify the classes at high risk by ranking the classes in software according to their fault contents. Therefore, fault forecasting method is used at design level to enhance the reliability by redesigning and reviewing the classes according to the rank assigned to them. In this way all the objectives of the present research are accomplished.

7.3 Testing the Hypotheses

Hypothesis testing is a process by which an analyst/scholar tests research questions/hypotheses. The methodology employed by the analyst depends on the nature of the data used, and the goals of the analysis. The goal is to either accept or reject the
hypothesis. Hypothesis testing is used to infer a result of a hypothesis performed on sample data. The scholar tested the hypotheses with her analysis results as follows

\[ H_1: \text{Internal metrics have significant relationship with external attributes of product;} \]

To prove or disprove the hypotheses the scholar provides inputs from the previous chapters so that it can give the base to test the hypotheses. For 1st hypotheses three parameters are provided and these are as follows.

*Table 4.4 to 4.7* provide the results of univariate regression analysis with regard to ungraded, high severity, medium severity and low severity number-of-faults per class for each predicator i.e. WMC, RFC, CBO, DIT, NOC, LCOM and SLOC. The \( R^2 \)-values from *table 4.4 to 4.7* show that all seven metrics have association with every category of faults except DIT which does not show association with high and medium severity faults. But the results of multivariate regression analysis and best model details with regard to ungraded, high severity, medium severity and low severity faults depicted in *tables 4.8 to 4.15* show that all seven metrics including DIT are taken for analysis are related to faults, or predict faults. The predictive power of each metric is different for each category of faults, shown through the graphical representation in *fig 4.8 to 4.14* for comparing the capability of each metric to predict the high, medium and low severity faults with the capability to predict the ungraded faults. Thus the seven internal metrics WMC, RFC, CBO, DIT, NOC, LCOM and SLOC metrics association with faults from static fault-report, which is an external attribute of product or in order words it prove that these have significance relationship with external attribute.

2. The proposed design metrics are theoretically analysed on the basis of elementary constituent of cognitive theory for object-oriented metrics to observe the relationship with faults induced in class. The four proposed design metrics are internal product metrics of design level. Faults are collected from test-report of project and is an external attribute of product. Analysis results are provided in *table 6.1* of chapter 6, show that faults induced in a class increases with increase in the value of CMCM, CICM and
CALM metrics. But CCOM metric shows inverse effect that means faults decrease with the increase in the value of CCOM.

It proves that internal metrics CMCM, CICM, CALM and CCOM have significant relationship with internal attributes i.e. faults induced in the class during its design and coding.

3. Proposed design metrics are also empirically analysed along with the cognitive analysis to observe the association with fault induced in a class having linear regression and curve-fit estimation for linear and non-linear mode of association and results are summarized in table 6.3 and 6.4. The $R^2$-value from these tables show that CMCM, CICM and CCOM metrics show both linear and non-linear mode of association with faults but CALM shows only linear mode of association with faults. The graphical representation of comparison of various curves in fig. 6.7 to 6.10 for each metric vs faults show linear and non-linear associations. Thus, empirical analysis also proves that there is significant relationship between proposed internal design metrics and faults that is an external attribute.

**H 2** There is a significant relationship among external links and induced faults in a class

To test this hypothesis the scholar has provided definition of all the four newly proposed metrics along with the theoretical basis as proposition criteria. These proposed metrics are Class Member Complexity Measure (CMCM), Class Inheritance Complexity Measure (CICM), Class Aggregation Level Measure (CALM) and Class Cohesion Measure (CCOM).

*Class Member Complexity Measure (CMCM)*

It measures class complexity through public/protected decelerations within class. The more number of public/protected declarations give rise to more external links of a class with other classes and more faults are induced. The value of $R^2$ and CMCM co-efficient from table 6.3 and graphical representation in fig. 6.7 corresponding to CMCM metric shows that this predictor have significant and positive relationship faults induced in a
class. Thus there is significant and positive association between external links and induced faults in the class.

**Class Inheritance Complexity Measure (CICM)**

It measures the class-complexity through inheritance. CICM value depends upon the number of immediate ancestors and their depth/level in the inheritance hierarchy along with the depth of class itself. So higher value of CICM means more number of immediate ancestors deeper in the hierarchy and/or deeper the class itself in the hierarchy, it means that the class is inherently associated with more number of ancestor classes. Thus, the class having more value of CICM is linked with more number of other classes or linked with more number of classes. The $R^2$-value and CICM co-efficient value given in table 6.3 and graphical representation in fig. 6.8 for CICM metric indicate that there is significant and positive association between this metric and faults. This prove that external links and faults in a class are significantly related with each other.

**Class Aggregation Level Measure (CALM)**

CALM measures the class complexity through aggregation level used within the class. The aggregation level means that how many attributes out of total attributes in a class are declared as another classes type or lesser defined type. So more value of CALM for a class shows that the class is using more of its capability to link with other classes through aggression. The $R^2$-value and co-efficient value of CALM from table 6.3 and curve slops in fig. 6.9 show that CALM metric has significant and positive association with faults. Therefore it proves $H_2$ i.e. external links with other classes and faults induced in a class are significantly related to each other.

**Class Cohesion Measure (CCOM)**

It measures the class complexity through class cohesiveness of class. More cohesive class has more dependency within class by more attributes sharing by the methods and least dependency or link with other classes. So more value of CCOM of a class means the class behaviour as more independent module and least eternal links to other classes. The
value of $R^2$ and CCOM co-efficient from table 6.3 and slope of curves in fig. 6.10 shows that CCOM metric have very little and inverse association with the faults from the test-report. The CCOM value has inverse association with eternal links with other classes of a given class. Thus external links and faults induced of class are directly related with other, but level of association is not significant. Hence, $H_2$ is disapproved.

Thus the 2nd hypothesis “There is a significant relationship among external links and induced faults in a class” for CMCM, CICM and CALM metrics proves and disproved for CCOM.

$H_3$ Proposed metrics predict faults at the design level of the object-oriented software;

Table 6.7 describes the analysis results of various regression models. These results show that model 1 includes all predictors and having maximum $R^2$ value but $R^2$-adj. value is not maximum for it. This indicates that all predictor terms are not effective and some may be excluded. The process of excluding the terms from model 2 to 6 depict that some model shows increase in the values of $R^2$-adj. and decrease in the $R^2$ values. By analyzing various models from 1 to 6 the best model excludes the cub. CMCM and Sq. CCOM and model 4 is chosen best. This model has maximum $R^2$-adj. value and better significance level of coefficients for each variable.

Table 6.8 and 6.9 show the summary and details of coefficient of model 4 respectively. The best fault prediction model is mathematically represented as

$$\text{Faults} = 0.126 (\text{Sq. CMCM}) - 1.140(\text{CMCM}) + 1.862(\text{CICM}) - 0.219(\text{Sq. CICM}) + 2.631(\text{CALM}) + 6.496(\text{CCOM}) - 2.112$$

The four proposed metrics can be evaluated by using the static information obtained from the design phase, so values of all terms taken as predictors in above model are evaluated at design phase. From Model summary in table 6.8 $R^2$ Value of model is 0.818 and significant level is 0.000 thus fault prediction model considered as best is able to predict nearly 82% of total faults with good accuracy at design phase using the predictors CMCM, Sq. CMCM, CICM, Sq. CICM, CALM and CCOM. The validation
process of faults prediction model using spearman’s rank correlation coefficient also support it.

So the 3rd hypotheses “Proposed metrics predict faults at the design level of the object-oriented software” hence proved.

7.4 Findings of the Study

The key findings of the study are as follows:

1. Traditional size metric, SLOC is found effective in object-oriented environment and also to predict the number-of-faults in a class using the public NASA Data set KC1.

2. The fault prediction capability of SLOC varies with the severity level of faults i.e. High, Medium and Low. The prediction capability is very less for high severity faults and similar and significant for medium and low severity faults.

3. The CK-metrics suite, which includes WMC, RFC, CBO, DIT, NOC and LCOM is also analyzed to predict the number-of-faults using Public NASA data-set and WMC, RFC and CBO are more effective than DIT, NOC and LCOM. But DIT and NOC along with SLOC are found the predictors of best chosen regression Model to predict total number-of-faults (ungraded i.e. without categorizing the faults as per their severity) in a class.

4. The WMC, RFC and CBO are found more significant as compared to DIT, NOC and LCOM for predicting the number-of-faults of each severity level (High, Medium and Low). The best Model chosen for predicting the high, medium and low severity faults includes different predictors, where as prediction capability of best medium severity fault model, low severity fault and high severity fault model are found in decreasing order.

5. The newly proposed design metrics were theoretically evaluated using formal set of mathematical properties introduced by Weyuker which are appropriate for object-oriented software metrics. The evaluation results are found satisfactory and thus these four metrics are considered as valid internal design metrics.
6. After evaluating against the five specific properties for complexity metrics given by Briand et al, it was observed that out of five, four properties are satisfied by all four design metrics; therefore these metrics are valid complexity design metrics.

7. The four design metrics are found valid to evaluate the complexity of object-oriented design through data-hiding, inheritance, aggregation (association) and cohesion concepts.

8. Theoretically analyzing four design metrics, using the basic constituents of cognitive-theory for object-oriented software metrics, it is found that these metrics shows interference effect, fan effect and familiarity due to the external association links of a class with other classes and have effect on the recall, comprehension and understanding of class. Memory-Span effect is not observed during this analysis.

9. The proposed CMCM, CICM and CALM metrics values shows direct association with the external links of a class with other classes, make the recall and comprehension of class difficult and hence they show positive association with cognitive-complexity or induced faults during design phase at class-level.

10. CCOM metrics value are inversely associated with external links of a class with other classes as more cohesive class have least link with other classes make the recall, comprehension and understanding of class simpler and this metric shows negative association with cognitive-complexity and induced faults at class-level.

11. It is observed empirically that all four proposed metrics have association with the faults in class collected from the test-report. CMCM, CICM and CALM shows significant and positive association, but CCOM shows less significant and negative association with the faults as shown in the results of univariate linear-regression analysis.

12. To explore the mode of association other than linear between independent and dependent variables curve-fit-estimation is used and found some non-linear mode of association i.e. quadratic and cubic are effective and are included as predictor to get the best multivariate model to predict the faults.
13. The model proposed for faults prediction includes CMCM, Sq. CMCM, CICM, Sq. CICM, CALM and CCOM as predictors and is a accurate model as its validity is checked through the spearman’s rank correlation coefficient analysis.

14. The prediction model is found useful in identifying the classes at high risk to be redesigned and reviewed to enhance some level of software reliability before the actual code is developed or at the end of the design phase.

7.5 Areas for Further Research

Object-oriented design metrics have become very important these days in the field of computer science and engineering. A sincere effort can be devoted to have metrics at pre-coded phase. This field of study enhances both quality and quantity of software design metrics research. The present study can further enhance the use of static data available at the design phase of object-oriented software development. The software industries/houses or academic institutions can get the benefit of the study for reducing the overall efforts in terms of manpower, money and time in producing reliable products by redesigning the classes after getting timely feedback and hence many disastrous incidents can be avoided. The proposed metrics measures the class complexity on the basis of data-hiding, inheritance, aggregation and cohesion design concepts. Thus, similar type of metrics can be devised by focusing on other design concepts of object-oriented paradigm to optimize the use of static information available up to design phase. The use of proposed metrics can be generalized by further validating them using the data from industrial projects developed in various object-oriented languages like java, C#, VC++ etc. The proposed metrics predict the total number of faults in class. However, the same study can be extended to predict the faults according to their severity level. Proposed metrics can further evaluate under different industrial and academic environments to redefine the design metrics according to specific industrial/academic needs to enhance reliability. The class-level faults prediction are further utilized/extended to estimate the total faults induced at system level. The enhancement in the level of reliability through classes redesign having specific number of faults predicted through faults-prediction model could be further quantified using the data from real-time applications for intensive
and early feedback to attain reliability. Further, it can attract the researchers at the universities, research labs with funded project. Many metrics can be devised for different purposes of software related work if funding agencies and governmental bodies sponsor the quality researches in this direction.

Finally, the scholar believes that these types of studies are needed to improve the quality and timely production.