The Impact of Module Attributes on Project Defect Density

Chapter
The Impact of Module Attributes on Project Defect Density

Syed Muhammad Ali Shah[1] **and Maurizio Morisio**

Abstract. Context- Software modules are the basic building blocks of any software project and these modules are engineered differently for different types of projects. Having a diversity of engineering practice, the attributes of these modules should have different impact on projects. Objective: We studied 54 software projects to analyze the impact of modules attributes on the project's quality in term of defect density (DD). Results: We found that the module's attributes i.e. very small modules on size and defect free modules have significant impact on the projects DD. The former more percentage resulted in higher projects DD and later more percentage resulted in lower projects DD. The attribute module dependencies have no significant impact on the projects DD. Moreover, we found that projects type (student, open source) having higher DD have more percentage of modules with higher DD, but this trend is not found in the close source projects. We found the significant relationship of projects DD with the module attributes (defect free and very small). Conclusion: Different module attributes have different impact on projects DD and modules behave differently for different types of projects. This empirical work suggests practitioners and researcher with evidence how module attributes affects the projects DD. The authors recommend some suggestions to take into account during the software construction.

Keywords: Module • Size • Quality • Defect Density.

[1] S. M. A. Shah (✉)
Politecnico di Torino, Corso Duca degli Abruzzi, 24 10129 Torino, Italy
e-mail: syed.shah@polito.it

M. Morisio
Politecnico di Torino, Corso Duca degli Abruzzi, 24 10129 Torino, Italy
e-mail: maurizio.morisio@polito.it

1. Introduction

The relationship between size and quality in software projects is highly debated, both at project and module level. Typically a module is intended at the physical level (a file as part of a project), its size measured in LOC, its quality measured in defects or defect density (DD defined as defects/size). In research studies (1)(2), it is found that smaller modules have higher DD compared to the larger modules. Many researchers have made their efforts to highlight different relationship between the size and DD of the modules. For example, in studies (3)(4) it is found that the modules DD increases with the increase in the size of the modules. In studies (1)(2) it is found that the DD decrease with the increase in the size of the modules. However the studies (5)(6) show no significant relation of DD with the size of the modules. In summary the studies show the increase or decrease of DD with size. Although at project level, the consequences of these observations that are proven at module level are not well known. For example if a project is constructed with larger sized modules or small sized modules then what should be the resultant DD of the project. This allows us to devise our research to study the module attributes that have influence on project DD. This paper studies the modules attributes and their effect at project level. The paper is organized as follows; Section 2 discusses the related work. Section 3 presents the design of the study. Section 4 presents the results. Finally, discussion and conclusions are presented in Section 5.

2. Related Work

Many studies have analyzed modules within a single project. Withrow (3) showed her work by examining the 362 modules of the ADA. She divided the modules into 8 groups based on the module size. She showed that after a certain range of module size (161-250 lines of code) the defects start increasing with the module size. This result also supports the Banker and Kemerer hypothesis (4) where they proposed the optimal module size. The minor size of the module has positive impacts and for greater size, the negative impact starts. Moller and Paulish highlighted that for the module of size smaller than 70 lines of code DD increases significantly and modules that have size greater than the 70 lines of code have similar trends toward DD (7). Hatton (8) studied 'NASA Goddard' project along with Withrow's data set. He classifies the modules in two categories. For size up to 200 LOC, he suggested that the DD grows logarithmically with the module size and for modules larger than 200 LoC, he observed a quadratic model. Rosenberg (9) has commented on Hatton (8) argument that the observed decrease in DD with rising module sizes is misleading.

Shen et al. (1), worked on three separate releases of an IBM software project by studying 108 modules. They highlighted 24 software modules with size exceeding 500 LOC. They affirm that increases in size did not influence the DD. For the remaining 84 modules, they showed that DD declines as size grows. A study done by Basili and Perricone (2) showed the division of 370 modules into 5 groups based on the module size with increment of 50. They observed the trend of having lower DD of larger module. Fenton and Ohlsson (5) have studied the modules of large telecommunication projects. They selected the modules randomly for the study but did not observe significant dependence of module size with DD. In addition many studies analyzed modules from more than one project. Andersson and Runeson (6) replicated the Fenton and Ohlsson (5) study using the data of three telecommunication projects. They were also not so much successful in finding the significance relation between the number of defects and LOC compared to the original study. El Emam et al. studied three software projects written in C++ and Java. They highlighted that there is a continuous relationship between the class size and faults (10). Koru et al. (11), studied four large open source projects: Mozilla, Cn3d, JBoss, and Eclipse. They observed that smaller classes are more problematic than larger ones. In particular, for open source software the theory of Relative Defect Proneness (RDP) (12) is postulated about the size defect relationship, stating that "smaller modules are less but proportionally more defects prone compared to larger modules"

In related work we found different types of relationship between the module size and DD. Although these relationships are important to understand, what is still unknown is the impact of these characteristics on the projects DD. For example it would be worth understanding the size or quality of module impact on overall project DD.

3. Research Design

In this section, we present the research questions and the selected projects on which the analysis is performed. The research questions are formalized from the related work, considering the mostly studied attributes (size, quality, dependencies) at modules level. The overall goal of this work is to understand the effect of module attributes on projects DD and how much it is different for different type of projects. Table 1 summarizes the formulated research questions and corresponding hypothesis of the study. We selected the last releases of 54 software projects from the "*Promise data repository*[2]*"* (13). The projects inclusion criterion was the availability of the required metrics (defect per module, no of line of code (LoC) of modules, module dependency metrics) to answer our research questions. The pro-

[2] http://promisedata.org/

jects did not satisfy the inclusion criteria were excluded. The data set contains 23 close source projects, 15 open source software projects and 17 are academic projects that were developed by the students.

Table 1. Research questions and hypothesis

RQ 1: What is the distribution of modules on size in projects? The goal here is to characterize the modules on size of different projects, and then check the following hypothesis .
RQ 2: What is the distribution of defect free modules in a project? The aim here is to find the difference in DD of projects by defect free modules, checking the following hypothesis H2.1: Projects have the same distribution of defect free modules. H2.2: Projects with lower DD have a larger percentage of defect free modules.
RQ 3: How modules dependencies affect the projects DD? The goal here is to find the influence of modules dependencies on the projects DD, checking the following hypothesis H3.1: Projects having higher dependencies of modules have higher DD. H3.2: Projects having lower dependencies of modules have lower DD.
RQ 4: How defect density of modules affects the defect density of projects? The goal here is to find the difference in DD of projects by modules DD, checking the following hypothesis. H4.1: Projects with more DD have a larger percentage of modules with higher DD.

Notion of Defects: In this work, we consider only post release defects, therefore temporary problems, non defect items like issues, warnings; temporary problems and further enhancements are not included. We believe on the authenticity and reliability of post release defects of the projects available at Promise repository as the data set is publicly available and used in many prior research studies.

Notion of Modules: In this work, the module is assumed to be a smallest unit of functionality i.e. set of declarations and subroutines usually belonging to one file.

Notion of Defect Density: DD is the key object of this study the DD is reported in LoCs on defects. For each project we successively extract and add all the defects related to each module, to obtain the total number of defects in a project. In a similar way (addition of all modules LoC) we calculated the total size of the project in LoC. To obtain the DD per thousand lines of code, we multiply it by 1000.

For the data analysis we used both graphical representation and the mathematical calculation “percentage”. The former gives us the immediate comparison and the later shows weight and the influence to characterize. Where applicable we also used the statistical methods for data analysis.

Concerning RQ1, we carried out the preliminary analysis of three types of projects (student, open source and close source) to categorize the projects in small, medium and large category using the *k* mean clustering algorithm. Afterward we find the percentage of distribution of very small modules in all categories of projects then we compare the percentage of modules and DD of projects. To find the statistical significant difference between the two groups we used the non parametric test Mann Whitney. For projects type student, we found 7 small projects, 7 medium and 3 large projects. For projects type open source, we found 6 small, 5 medium and 4 large projects. In projects type close source, we found 14 small, 3 medium and 6 large projects. Table 2 shows the three categories of software projects i.e. small, medium and large it shows that the large projects in term of size have lower DD in all types. We clustered the modules of each type of projects into 5 categories (very small, small, medium, large, and very large) based on size using the *k* mean clustering algorithm. We observed that the mean DD of very large modules is less than the DD of other categories of modules.

Table 2. Categories of software's in term of size

Type	Category	No of Projects	Avg size in LoC	Avg DD [KLOC]
	Small	7	4350	4.55
Student	Medium	7	12079	1.4
	Large	3	40984	1.2
	Small	6	28639	6.15
Open	Medium	5	114838	5.27
Source	Large	4	285061	1.22
	Small	14	12240	4.67
Close	Medium	3	70784	2.38
Source	Large	6	437029	1.6

Table 3 reports the DD of very small, small, medium, large and very large modules of each type of project. The preliminary observation shows that in all types of projects there is a smaller percentage of very large modules.

Concerning RQ 2: We first extract those modules that are defect free and then find out the percentage of these defect free modules in a project. For the second hypothesis we used a k mean clustering algorithm to cluster the projects based on DD and then performed the analysis observing the DD and percentage of defect free modules.

Concerning RQ 3: To find the module dependencies we used two metrics suggested by Martin[3] to calculate the module dependencies. The metrics we used are:

[3] http://www.objectmentor.com/resources/articles/oodmetrc.pdf

Afferent Coupling (AC): The number of modules that depend on M.

Efferent Coupling (EC): The number of modules that M depends upon.

We first extracted the dependencies of each module using the above defined two metrics. Afterwards we average the module dependencies of each project to find average module dependencies of a project. Then we performed the analysis observing, how DD of projects is affected by the module dependencies using a statistical measure regression analysis.

Table 3. DD of different categories of modules

Type	Category	Range in Loc	No of Module	Avg DD	% of Module
Student	V Small	1 - 205	640	7.22	67.4
	Small	206 - 565	204	1.6	21.4
	Medium	575 - 1250	78	1.15	8.2
	Large	1347 - 2781	22	0.77	2.3
	V Large	3211- 5924	5	1.48	0.5
Open Source	V Small	1-283	4732	33.2	73.6
	Small	284 - 850	1207	2.86	18.7
	Medium	853 - 1902	329	2.06	5.1
	Large	1940 – 4114	122	1.38	1.9
	V Large	4202- 3175	32	0.6	0.5
Close Source	V Small	1-132	30179	5.6	82
	Small	133-465	5209	1.39	14.2
	Medium	466-1263	1065	0.77	2.9
	Large	1266-2927	222	0.66	0.6
	V Large	2928 – 9878	40	0.61	0.1

Concerning RQ4: For the analysis we consider only top five projects with higher DD. Consequently we observe the distribution of percentage of modules in these projects that have higher DD.

4. Results

RQ 1: What is the distribution of modules on size in projects?

From Table 3 we found that in all three categories of projects the distribution of modules on size is very different. Hence the distribution of modules on size is very different in all categories of projects; we can reject our hypothesis H1.1, that the projects have not same distribution of modules on size. Considering hypothesis H1.2, we found that very large module have very small percentage in all types of

projects. For projects (student, open source) it counts 0.5%, and for close source projects the percentage is 0.1%. Hence we can reject our hypothesis H1.2, that the projects have not more percentage of large modules. The secondary observation we obtain is the higher percentage of very small modules in all types of projects. It is above 60%. Figure 1 shows the box plots of categories of very small modules in all categories of projects. In all categories of projects, we found the distribution of very small modules of large sized project less than the small and medium sized projects. We notice that there is more variation of percentage of very small module in student projects as compared to open and close source projects.

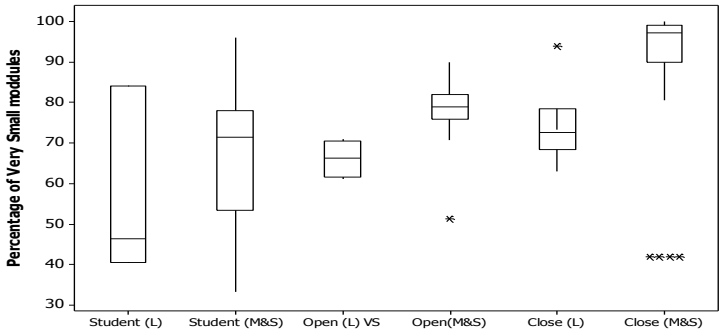

Figure 1 Percentage of Very Small modules in projects

Afterward we compare the distribution of very small modules and DD of large sized projects, with the distribution of very small modules and DD of small and medium sized projects. Table 4 reports the distribution of very small modules and DD of large sized projects compared to small and medium sized projects. It shows that there is a smaller percentage of very small modules in the larger sized projects compared to the small and medium sized projects of all types.

Table 4. Distribution of very small modules and DD in large sized projects vs. small and medium sized projects

Type	% of VS modules on Large Projects	DD of Large Projects	% of VS modules in Small & Medium Projects	DD of Small & Medium Projects
Student	57	1.23	67	2.98
Open Source	66	1.22	77.6	5.7
Close Source	74.3	1.6	91.2	4.27

We used Mann Whitney test to see the significant difference of percentage of very small modules between large sized projects with small and medium sized projects. We obtained p value = 0.0068 which is below the significant value 0.05. This con-

firms that the distribution of very small modules in large projects is different compared to small and medium size project in all types. Recalling Table 2 we had observed that large projects have lower DD and hereinafter we found that projects having more percentage of very small modules have higher DD. In particular, when we looked for a comparison between the large sized projects with small and medium sized projects, we found that large projects have a smaller percentage of very small modules and their corresponding DD is also lower than small and medium size projects. Similarly if the projects are constructed by the higher percentage of very small modules then the overall DD of the project is higher. Based on the empirical evidence one might can state that large projects have a smaller percentage of very small modules that would result in lower DD of large projects. However there could be many other factors affecting the DD of larger projects e.g. The larger project is normally taken more seriously, have rigorous testing etc.

RQ 2: What is the distribution of defect free modules in a project?

We found that the distribution of defect free modules in different types and categories of projects is very different. The difference of distribution allows us to reject our formulated hypothesis H2.1. Table 5 reports the distribution of defect free modules in each category of projects by type. Comparing overall by category, it shows that the large projects of all types have a higher number of defect free modules compared to medium and small category of projects and they make a larger percentage of code size as well. Similarly comparing the defect free modules by type, we found that close source projects have more percentage of defect free modules than open source and student projects. The defect free modules found in close source, open source and student projects are 81%, 59% and 69.3% respectively. Thus it shows that there is more quality attention given to the units of large projects compared to the medium and small sized projects. In addition the close source projects have more percentage of defect free modules compared to open source and student projects. This shows that close source projects have better construction quality.

Table 5. Distribution of defect free modules and percentage of code of defect free

Type	Category	% of Defect Free Modules	% of code of Defect free modules
	Small	59	38.1
Student	Medium	70.4	41.1
	Large	78.5	56
	Small	56	39.7
Open	Medium	54	35.4
Source	Large	67	53.7
	Small	81.4	70.7
Close	Medium	75.3	55.7
Source	Large	87.6	75.3

Concerning hypothesis H2.2, we performed the regression analysis to understand the relationship of defect free modules with the projects DD. For student and open source projects we obtain the R^2 = 38.4% and 54.7% respectively having a partial impact. However for close source projects we obtain R^2 = 5.9% having a very limited impact. To test our hypothesis H2.2 we cluster the projects based on the DD using *k* means clustering algorithm for all types of projects. Table 6 reports the DD and percentage of defect free modules in projects. In all types of projects we found that the projects having a higher percentage of defect free modules have lower DD compared to projects having a smaller percentage of defect free modules. Thus we accept our hypothesis H2.2 that project with lower DD have a larger percentage of the defect free modules.

Table 6 Projects Defect density Vs % defect free modules

Type	No of Projects	Defect Density	% of Defect Free
	4	5.8	41.2
Student	5	2.3	61
	8	1.2	84
	6	8.8	22.2
Open Source	5	4.54	58.2
	4	0.5	92.7
	13	4.3	82.3
Close Source	3	4.0	56.3
	7	1.93	93.28

RQ 3: How modules dependencies affect the projects DD?

In our data set the module dependency metrics are reported for 36 projects. We answer RQ3 considering those 36 projects, in which there are 17 students, 14 open source and 5 close source projects. Table 7 reports the average module dependencies of each category of project along with the DD. Observing the Table 7 we can't find any difference of module dependencies and the projects DD in different categories of projects. This makes us to perform our analysis only on the project types, to understand the impact of module dependencies on the projects DD.

Student projects: The obtained R^2 value is found to be 27.7%. There is a positive correlation but it has a limited practical impact. The regression equation is:

$$DD = 3.11 - 1.10AC + 0.825EC$$

Open source projects: The obtained R^2 value is found to be 10.6% having limited practical impact. The regression equation is:

$$DD = 10.9 - 0.582\ AC - 0.564\ EC$$

Close source projects: The obtained R^2 value is found to be 60.5% having a significant practical impact. The regression equation is:

$$DD = - 1.52 + 0.889AC + 0.012EC$$

Table 7. Projects module dependencies and DD

Type	Category	Project	Afferent coupling	Efferent coupling	Defect Density
	Small	7	2.5	4.0	4.55
Student	Medium	7	4.2	3.8	1.4
	Large	3	4.7	6.1	1.2
	Small	6	3.4	6.0	6.15
Open Source	Medium	5	5.3	5.0	5.27
	Large	3	6.6	6.56	4.24
Close Source	Large	5	2.6	12.2	1.01

Considering all the observation, we do not find any significant impact of modules dependencies on the projects DD in our sample of data. The relationship of module dependencies and DD of projects in type (student, open source) is found to be limited, however considering close source projects there is some practical impact. Thus we cannot accept or reject our formulated hypothesis H3.1 and H3.2.

RQ 4: How defect density of modules affects the defect density of projects?

To answer the RQ4, we selected top 5 projects from each type that have higher DD. Table 8 reports the projects having higher DD in each type. For projects (student, open source) the average percentage of module with higher DD is 51% and 59 % respectively. On the contrary for the close source projects the average percentage of module with higher DD is only 19.6%. Thus our formulated hypothesis is accepted for the students and open source projects that the projects with more DD have the more percentage of modules with higher DD; however the hypothesis H4.1 was not found true in the close source projects.

Table 8 Projects with higher DD vs. modules with higher DD

Type	Projects Avg DD	Projects DD	% of modules with higher DD	Avg % of module with higher DD
		11.5	56	
		5.1	66.6	
Student	5.53	4.8	51.2	51
		3.4	45.5	
		2.7	35.7	
		15.6	50	
Open	10.11	13.0	91.7	59

Source		11.3	74.3	
		6.1	60	
		4.4	20	
		8.0	35.2	
Close	6.69	7.2	14.7	19.6
Source		6.2	21.6	
		6.0	12.7	
		6.0	14	

4.1 Threats To Validity

We discuss in this section validity threat using the classification proposed by (14).

Internal validity: In this study we only focus our observation towards some basic module attributes like size, quality, dependencies etc., to find their impact on projects DD. However there are many other module attributes that should have an influence on projects DD e.g. testing effort, testers experience and testing techniques etc. We acknowledge all other module attributes but for this study we only focus on the studied ones. Construct validity: In this research, we are dependent on the data logs provided by the *Promise data repository*. Surely, some potential concerns can be raised about the given data set e.g. how many modules may be left, how many defects may be raised and fixed before data collection and how many defects may not be recorded in logs etc. We consider this threat, but as the data set is publicly available and has been used in many previous studies, we believe its authenticity. External validity: In this study our findings are based on a small set of projects i.e. 54 software projects of different nature considering the impact of only a few attributes. Although this number is small but not negligible, this adds to the confidence by presenting some module attributes and their impact on projects DD.

5. Discussion And Conclusion

This study has two folded outcomes, first how different the internal structural properties of three types (student, opens source, close source) of projects, secondly how module attributes affects the quality (in term of DD) of projects. The results show that the module attributes have some impact on projects DD. We found that the projects have not the same distribution of modules on size. In all types of projects there is a very small percentage of very large modules. The percentage of

very small modules are less in large projects compared to medium and small sized projects. The empirical evidence shows that DD of the project increases with more percentage of very small modules (RQ1). The quality of the module has significant impact on the project quality (RQ2). We found that the module dependencies have not significant influence on the projects DD for student and open source projects; however module dependencies have some impact on close source projects DD. Having only 5 close source projects for the analysis does not add much confidence in the results (RQ3). We found that it is not always true that modules with higher DD would result in higher DD of projects as it is found true for student and open source projects but not for close source projects (RQ 4). The projects DD can be predicted by using the modules attributes (percentage of very small and defect free modules) as the significant relationship between projects DD and attributes is found (RQ5). Authors want to give some suggestions to practitioners aimed to assess their projects based on our empirical findings.

(1) More percentage of very small modules affect negatively to the projects DD.

(2) Module compositions have not much effect on projects DD.

(3) Modules DD may not always be significant to predict the projects DD.

(4) The module attributes (% of very small modules, % of defect free modules) can be used to access the projects DD.

The authors have confidence in these results. These are based on the validated data set of projects that have been used previously and available for research purposes at Promise *Repository*. The artifacts of this study also give direction to the researchers, that different types of projects have different internal structure and their characteristics are quite different from each other. In addition it also shows the importance regarding the previously conducted studies (where mostly the relationship of different module attributes has been shown) but their impact on overall project was unknown. Therefore, we recommend researcher to study more module attributes and their impact on projects. We think that the main limitation of this study is the very few projects and module attributes under study. More attributes like (testing efforts, development process, testing techniques, experience of the team) should be studied to find their impact on the projects DD. As a future work we will expand our research to find another module attributes that have an impact on project DD.

6. References

1. Shen VY, Tze-jie Yu, Thebaut SM, Paulsen LR. Identifying Error-Prone Software—An Empirical Study. Software Engineering, IEEE Transactions on DOI - 10.1109/TSE.1985.232222. 1985;SE-11(4):317–24.
2. Basili VR, Perricone BT. Software errors and complexity: an empirical investigation0. Commun. ACM. 1984;27(1):42–52.
3. Withrow C. Error density and size in Ada software. Software, IEEE DOI - 10.1109/52.43046. 1990;7(1):26–30.
4. Banker RD, Kemerer CF. Scale Economies in New Software Development. Software Engineering, IEEE Transactions on DOI - 10.1109/TSE.1989.559768. 1989;15(10):1199–205.
5. Fenton NE, Ohlsson N. Quantitative analysis of faults and failures in a complex software system. Software Engineering, IEEE Transactions on DOI - 10.1109/32.879815. 2000;26(8):797–814.
6. Andersson C, Runeson P. A Replicated Quantitative Analysis of Fault Distributions in Complex Software Systems. Software Engineering, IEEE Transactions on DOI - 10.1109/TSE.2007.1005. 2007;33(5):273–86.
7. Moller K-H, Paulish DJ. An empirical investigation of software fault distribution. Software Metrics Symposium, 1993. Proceedings., First International. 1993. p. 82–90.
8. Hatton L. Reexamining the fault density component size connection. Software, IEEE DOI - 10.1109/52.582978. 1997;14(2):89–97.
9. Rosenberg J. Some misconceptions about lines of code. Software Metrics Symposium, 1997. Proceedings., Fourth International. 1997. p. 137–42.
10. El Emam K, Benlarbi S, Goel N, Melo W, Lounis H, Rai SN. The optimal class size for object-oriented software. Software Engineering, IEEE Transactions on DOI - 10.1109/TSE.2002.1000452. 2002;28(5):494–509.
11. Koru AG, Dongsong Zhang, El Emam K, Hongfang Liu. An Investigation into the Functional Form of the Size-Defect Relationship for Software Modules. Software Engineering, IEEE Transactions on DOI - 10.1109/TSE.2008.90. 2009;35(2):293–304.
12. Koru A, Emam KE, Zhang D, Liu H, Mathew D. Theory of relative defect proneness. Empirical Softw. Engg. 2008;13(5):473–98.
13. Boetticher G, Menzies T, Ostrand T. PROMISE Repository of empirical software engineering data [Internet]. Available from: http://promisedata.org/ repository, West Virginia University, Department of Computer Science, 2007
14. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslen A. Experimentation in software engineering: an introduction. Norwell, Massachusetts. USA: Kluwer Academic Publishers; 2000.