Toward An Empirical Study to investigate the size-defect Relationship Using ISBSG Repository

Fadi Wedyan
Department of Software Engineering
Zarka, 13315 Jordan
Fadi.wedyan@hu.edu.jo

Shirin Al-Manai
Department of Software Engineering
Zarka, 13315 Jordan
Sherine_almnay@yahoo.com

Wajeeha Al-Ajlouni
Department of Software Engineering
Zarka, 13315 Jordan
wajeehaajlouni@itc.hu.edu.jo

ABSTRACT
It is a common believe in the software quality community that software size is positively related to the number of defects (i.e., the larger the software gets, the more defects it contains). However, empirical studies in software engineering are influenced by many factors including the source of the benchmarks which makes comparing results relatively hard. In this paper, we present the result of an empirical study to explore the relationship between the software project size and defect using ISBSG data repository. The ISBG repository contains variety of benchmarks with different characteristics which makes it a well standard repository. We performed our study on 118 Java projects. We measured size with the IFPUG4+ function point metric. Our results show weak correlation between size and defects. However, for development projects (i.e., projects that are in the development phase), the correlation is stronger. Moreover, in mature projects, the correlation becomes very weak. These findings can be useful guide for more in depth studies to explore the size-defect relationship in large, object-oriented software.

Keywords
Software quality, size-defect relationship, ISBSG repository, software size, function points, IFPUG4+, empirical study.

1. INTRODUCTION
Ensuring software quality is a software process applied on the system to give measurable values to decide if the system meets the customer needs and requirements and help developers define defects in any phase (e.g. in-process, pre-release, and post-release defect). Software quality become more important as our dependent on computers is rapidly increasing. Therefore, requiring more reliable, dependable, and robust systems that must be developed fast, and account for future changes.

Empirical studies play a vital rule in software engineering. For such studies, the availability of benchmarks with quality attributes is essential. The International Software Benchmarking Standard (ISBSG) [1] repository contains wide range of projects collected from different environments using different methodological and techniques. The database is becoming a standard for empirical studies in software engineering.

IPAC '15, November 23-25, 2015, Batna, Algeria
© 2015 ACM. ISBN 978-1-4503-3458-7/15/11…$15.00
DOI: http://dx.doi.org/10.1145/2816839.2816852

Size is one of the important measures of software. It is commonly believed that size is related to the number of defects in software. That is, large modules are more error prone than small modules. While the size-defect relationship has been studies extensively, it is still not clear whether the size is the major factor of deciding which software module contains more faults. This is because of the many factors that play a role in error proneness of a module.

Software size is an internal attribute that is measured various metrics including Lines of Code (LOC), which is widely used mainly because the simplicity of computing it. LOC can be computed by counting lines of code in source code, executable lines with comments, or executable lines with data definition [3]. However, LOC has been criticized for being a rough metric. That is, LOC method does not give an accurate estimation for development process [4]. The metric is also dependent on the programming language, which makes results obtained using LOC hard to generalize. Moreover, the metric cannot be used during the analysis and design phases.

Function point was introduced by Albrecht [33] as a measure of the functional size of software. Function points (FP) measure software size by the functionality provided to the user based on logical design and functional specification. Unlike LOC, function points can be used to estimate software size in analysis and design phases. In addition, FP's are independent of the programming language or tools used for implementing the software.

In this paper, we present the results of an empirical investigation of the size-defect relationship. The study is performed using 118 Java projects from the ISBSG (R12) repository. In the study, we used FP's as a measure of the software size and the number of defects reported in the ISBSG repository. We used FP's to minimize the effect of the programming language used to implement the software and the effect of using different counting procedure. Projects used in the study cover variant domains and are collected from different resources, built by different developers, and in different times, varying from the 90's to current date. The study is the first build stone in a larger and a comprehensive empirical study that aims at analyzing the size defect relationship from different aspects. Specifically, the study reported in this paper aims to answer the following research questions:

Q1: Is there a relationship between the size of the project and the number of defects?

Q2: Does the project development cycle have an effect on the relationship between project size and the number of defects?

Our results show weak correlation between size and defects. This result confirms with some of the previously reported empirical

results. While some might argue that a module with larger size can contain more defects, because of many reasons (complexity of the code, data-flow complexity, etc.), the results do not confirm that. The complexity of the code, and therefore the chance of having more defects, is not necessary caused by the size.

To answer the second question, we evaluated two types of projects in the ISBSG repository, these are: (1) development projects, these are projects in the development stage, and (2) enhancement projects, which are projects in the maintenance phase. Our results show that there is a medium correlation between size and defects in development projects and a very weak correlation between size and defects in enhancement projects.

The rest of this paper is organized as follows. Section 2 summarizes related work in evaluating size-defect relationship. Section 3 gives a brief description of the ISBSG repository. Section 4 describes how data is collected and prepared for the empirical study. The results of the empirical study are given in Section 5. Section 6 describes the threats to validity. Finally, conclusions and future work are discussed in Section 7.

2. RELATED WORK

The size-defect relationship has been studied by many researchers. These studies can be classified into two categories: studies that explore the relationship between size, measured with various metrics, and defect, and studies that explore the relationship between size and defect density [24].

Akiyama [12] proposed the first model for size-defect relationship based on data collected from an assembly-language program for nine modules. Akiyama built a linear model for the size-defect relationship. However, the functional form of the relationship was not explored and the number of data points was small. Funami and Halstead [13] analyzed data presented by Akiyama and built Halstead's software science metric [14] measures size-defect relation and gives very high correlation between them in linear form.

Furthermore, Ottenstein [15] reported the linear relationship between size and defect based on a more appropriate model because it is built by comparing Halstead's software measures for different model's. In addition, Shen et al. [16] also built linear regression models by Halstead's software.

Compton and Withrow [17] proposed polynomial regression model to predict the number of the defect in any software system using the LOC as a size measure. They proposed the "goldilocks principle" which suggests that there be an optimal module size for minimizing defect density. Also, Schneidewind and Hoffman [18], Khoshgoftaar and Seliya [19], observed the high correlation between LOC and defects.

A relatively large number of studies evaluated the relationship between size and defect density (e.g., [6,7,8]). Defect density measures the percentage of faults in a software module and is computed by dividing the total number of defects by the size of the software [20] [23]. Basili and Perricone [21] study show that defect density was lower in large modules. Similarly, Moller and Paulish [22] observe that defect density is higher in small modules. Compton and Withrow [11] suggested that there be an optimal module size that would minimize the defect density. Hatton [27] results show that developers need to produce modules of intermediate size.

3. DATASET DESCRIPTION

The International Software Benchmarking Standards Group (ISBSG) is an organization that aims to "develop the profession of software measurement by establishing a common vocabulary and understanding of terms" [1]. The ISBSG data repository help researchers to achieve better understanding for the projects presented from various countries by giving more specific and details information. The ISBSG data repository (Release 12) contains 6,006 projects with different attributes give more specific details and information about project. Projects in ISBSG repository are collected from 26 different countries divided as : 32% from United States, 13% from Australia, 12% from Japan, 9% from Finland, 7% From France , 6% from Netherlands, 5% from India, 5% from Canada, and 4% From Spain.

The size of project is measured by function point using IFPUG4+, IFPUG, COSMIC, FiSMA, and NESMA. In addition, the older versions of these approach and others such as: Mark II, IFPUG (IFPUGE 2, IFPUGE 3), Furthermore these versions have "relative size" as a sizing attribute which divides size into intervals such as: XXS, XS, S, M, L, XL, XXL, and XXXL.

The ISBSG projects are developed in two types of programming languages: (1) 4th Generation languages (4GL), such as: Oracle, .Net, SQL, ABAP, NATURAL, Delphi, Access, ASP, and PowerBuilder, (2) 3rd Generation languages including projects developed in Java, and C/C++, and Visual Basic.

4. DATA PREPARATION

Having 6006 projects in the ISBSG, filtering is necessary to select projects that better fit our study. In the ISBSG repository, each project has a field called "Total Defects Delivered" that gives the number of known defects in the project. Project size is reported using various function points as described earlier. Furthermore, projects in the ISBSG repository are rated depending on the quality of the metadata given about a project. Rating varies from very good (rate A), to Unreliable (rate D).

In order to select suitable projects for our study, we applied the following steps, which were described in details in [26] [27]:

1- Select projects which have high data quality rated as 'A' and 'B'.

2- Select the projects sized according to the same functional sizing method.

3- Select the projects have same programming language.

4- Remove projects have blank fields, zero value and unspecified value like using, some, many, etc.

Applying step 1, we get 5558 projects with data quality A or B. The result of applying step 2 is shown in Table 1. In the table, column one shows the function size method, column two shows the number of project for which the function size method is used to measure the size. As shown in the table, metric IFPUG4+ is the most used functional sizing method (4082 projects).

Table 1. Functional Size Method

Functional size method	No. Projects
COSMIC	393
FISMA	528
IFPUG4+	4082
IFPUG old	210
NESMA	156
LOC	164
Mark II	19
Albrecht	2
Feature points.	2
Fuzzy logic.	2
Total	5558

Table 2. Distribution of programming languages used in implementing the selected projects

Programming Language	Project Number
.Net	123
C	183
C++	155
COBOL	558
Java	668
Oracle	117
PL/I	194
SQL	125
Visual Basic	310
ABAP	76
C#	60
COOL:Gen	56
Other	1457
Total	4082

Table 3. Distribution of Java projects with zero, unspecified, and defect value

Project Attribute	No. Projects
Project with zero defect value	194
Projects with unspecified value	356
Project with defect value	118
Total	668

We applied step 3 on the remaining 4082 projects filtered in step 2. The results are shown in Table 2. In the table, the first column shows the programming language used to implement the project while the second column shows the number of projects. As shown in the table, Java is the most used programming language with 668 projects. Therefore, we picked up projects implemented in Java. Finally, we applied step 4 on the Java projects, removing projects with zero defect value, and projects with unspecified values. The result of filtering projects by applying step 4 is shown in in Table 3. There are 118 projects remaining. These projects are implemented with the same programming language, measured with and the same functional size method, and have reported value for the number of defects.

5. RESULTS

In this section, we report the results of the empirical evaluation we performed in order to answer the research questions. Our study is performed on the 118 Java projects from the ISBSG data repository we collected and statistical analysis is performed using the SPSS [32] package.

Figure 1 shows the relationship between the size of the project, taken over all 118 projects, and the number of defects (regardless of the defect type). The figure shows that there is a weak relationship having a regression coefficient with a value of 0.077. This confirms with some of earlier reported studies that were performed using different metrics for measuring the size and on different data sets.

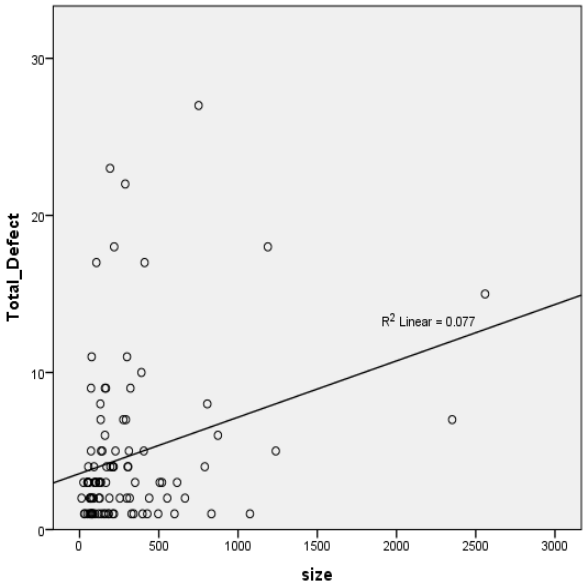

Figure 1. Size and defect relationship

In order 2 answer question 2, we studied the relationship between project size and each type of projects (development, enhancement). In the 118 Java projects we collected, there are 35 development projects and 83 enhancement projects.

Figure 2 shows the relationship between the project size and development projects. The results show a medium correlation between project size and number of defects (regression coefficient value of 0.145). However, the correlation is not strong and can be related to other attributes. Further investigation is needed.

The relationship between project type and the number of defects in enhancement projects is shown in figure 3. The results show a very weak correlation with a regression coefficient value of 0.006. Compared with development projects, these results can be explained by having more tests performed on enhancement projects, and therefore, many of the defects were revealed.

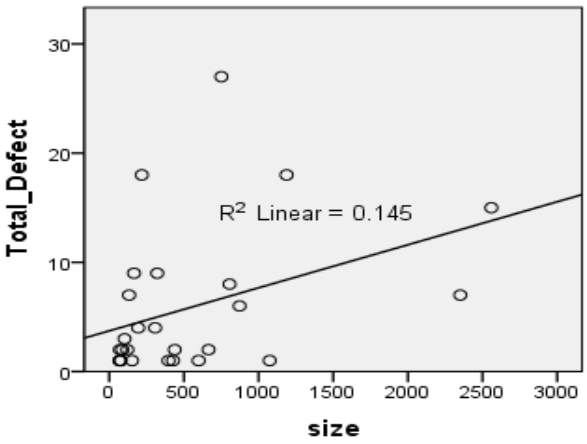

Figure 2. Size and Defect relationship in development projects

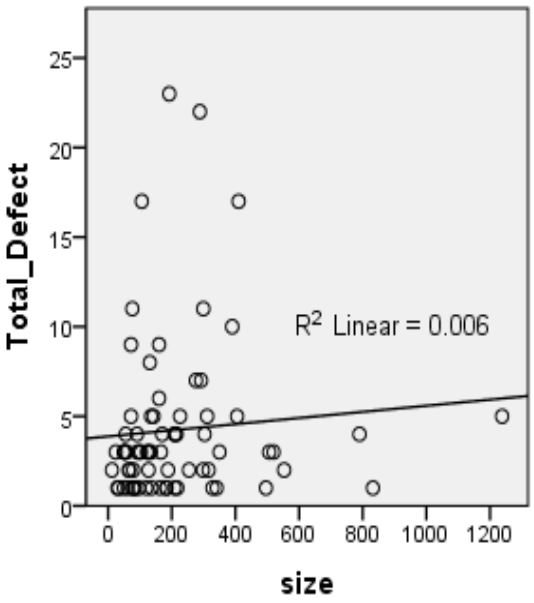

Figure 3. Size and Total Defect relationship in enhancement projects.

6. THREATS TO VALIDTY

Empirical studies are known to have limitations [29]. We identify three types of threats to the validity of this empirical study: construct validity, content validity, internal validity and external validity. Construct validity refers to the meaningfulness of measurements [30]. Notable construct validity is the inherent subjectivity of counting functional size measurement methods [9]. However, by using projects from the ISBSG repository with two highest categories of data quality minimizes the effect of this threat.

Internal validity is concerned with cause and effect relationships, the extent to which the changes in dependent variables can be stated to be caused by changes in independent variables. In our study, we identify one threat to the internal validity. It might be that the increase in the number of faults is affected by other factors, other than the size such as the maturity of the project, the testing approach, and the skills of the developers. However, since we used a large number of projects from a large database, the internal validity is endures to be fairly good. Moreover, we filtered out projects with low data quality, which make the results based on high quality projects.

External validity refers to how well the study results can be generalized outside the scope of the study [31]. The external validity of this study is limited to the use of projects implemented in one programming language (Java). There is no evidence that the results can be extended or generalized to other projects (even for other Java projects). The effect of this factor is limited since the projects in the studied sample come from different domains, with varied number of developers and with variant sizes and with varied attributes. In future work, we plan to study additional projects, implemented in other languages in order to confirm our results.

7. CONCLUSIONS AND FUTURE WORK

Understanding the size-defect relationship is important for software developing and testing. It helps developers “sizing” the modules in a way that decreases defects. While it is a common believe that larger modules contain more defects, empirical results do not confirm that.

There are many factors that can affect the size-defect relationship due to confounding effect [34]. Therefore, it is important to explore some of the attributes that might potentially affect the relationship. In this study, we explored the effect of the project development cycle. We used projects described in the ISBSG repository as development, which refers to projects in the developing phase, or enhancement, which refers to projects in the maintenance phase. Our results show that the size defect relationship is clearer in development projects, while the relationship is very weak in enhancement projects. There are many explanations for these results. For example, the effect of testing on revealing faults (i.e., enhancement projects passed more tests than development projects). However, further investigation is needed to confirm.

In future work, we plan to study software written in other programming languages such as C/C++. Also, we plan to identify the types of faults that are more affected by the size. The study will also be extended to identify the effect of the domain of the project om the size-defect relationship.

8. REFERENCES

[1] ISBSG, *Glossary of Terms*, V5.9.1, International Software Benchmarking Standards Group 28/02/2006

[2] ISO/IEC, " ISO/IEC 15026-1; 2013 systems and software engineering – systems and software assurance part1: concepts and vocabulary " ed: ISO/IEC .2013.

[3] C. Jones, “Programming Productivity”, New York: McGraw-Hill, 1986.

[4] J. Zhizhong, N. Peter, and J. Binghua, "The effects of software size on development effort and software quality." International Journal of Computer and Information Science and Engineering, Vol. 1, No. 4, pp. 230-234, 2007.

[5] S. Chidamber and C. Kemerer, “A metrics suite for object oriented design,” IEEE Trans. Software Eng., vol. 20(6), pp. 476-493, 1994.

[6] T.Compton, and C.Withrow,"Prediction and Control of Ada Software Defects, *J. Systems and Software*, vol. 12, pp. 199-207, 1990

[7] T. M. Khoshgoftaar and N. Seliya, The Necessity of Assuring Quality in Software Measurement Data , Proc. 10th Int’l Symp Software Metrics (METRICS’04), IEEE Press, pp. 119–130, 2004.

[8] T. Menzies, J. DiStefano, A. Orrego, and R. Chapman, Assessing Predictors of Software Defects, Proc Workshop Predictive Software Models, 2004.

[9] L. Lavazza, and R. Meli, "An Evaluation of Simple Function Point as a Replacement of IFPUG Function Point." the International Conference on Software Process and Product Measurement (IWSM-MENSURA, pp. 196-206, 2014.

[10] ISBSG, Data Collection Questionnaire New Department Redevelopment or Enhancement Sized Using COSMIC-FFP Function Points, version5.9.1, International Software Benchmarking Standards Group, 28/02/200628/02/2006.

[11] ISBSG, Data Collection Questionnaire New development, Redevelopment or Enhancement Sized Using COSMIC-FFP Function Points, version 5.9, International Software Benchmarking Standards Group, 24/05/2005.

[12] F. Akiyama, “An Example of Software System Debuggings,” Proc. Int’l Federation of Information Processing Societies Congress, vol. 1, pp. 353-359, 1971.

[13] Y. Funami and M.H. Halstead, “A Software Physics Analysis of Akiyama’s Debugging Data,” Proc. MRI XXIV Int’l Symp. Computer Software Eng., pp. 133-138, 1976.

[14] M.H. Halstead, Elements of Software Science. Elsevier, 1977.

[15] L.M. Ottenstein, “Quantitative Estimates of Debugging Requirements”, IEEE Trans. Software Eng., vol. 5, no. 5, pp. 504-514, 1979.

[16] V.Y. Shen, T.J. Yu, S.M. Thebaut, and L. Paulsen, “Identifying Error-Prone Software—An Empirical Study”, IEEE Trans. Software Eng., vol. 11, no. 4, pp. 317-324, 1985.

[17] B.T. Compton and C. Withrow, “Prediction and Control of Ada Software Defects”, J. Systems and Software, vol. 12, no. 3, pp. 199- 207, 1990.

[18] N.F. Schneidewind and H.-M. Hoffmann, “An Experiment in Software Error Data Collection and Analysis,” IEEE Trans. Software Eng., vol. 5, no. 3, pp. 276-285, 1978.

[19] T.M. Khoshgoftaar and N. Seliya, The Necessity of Assuring Quality in Software Measurement Data, Proc. 10th Int’l Symp. Software Metrics (METRICS’04), IEEE Press, pp. 119–130, 2004.

[20] L. Westfall, “Defect Density,” Available: http://www.westfallteam.com/Papers/defect_density.pdf. [Accessed: Feb. 2, 2015].

[21] Y. Funami and M.H. Halstead, “A Software Physics Analysis of Akiyama’s Debugging Data”, Proc. MRI XXIV Int’l Symp. Computer Software Eng., pp. 133-138, 1976.

[22] K. Moller and D. Paulish, “An Empirical Investigation of Software Fault Distribution”, Proc. First Int’l Software Metrics Symp., pp. 82-90, May 1993.

[23] C. Rahmani and D. Khazanchi, "A study on defect density of open source software", 9th IEEE/ACIS International Conference on Computer and Information Science, pp. 679-683, 2010.

[24] A.G. Koru , D. Zhang , K. El Emam, and H. Liu, "An Investigation into the functional form of the size-defect Relationship for software modules", IEEE transactions on Software engineering, vol. 35, No. 2, pp. 293-304.

[25] H. Zhang, "An investigation of the relationships between lines of code and defects”, In ICSM 2009, IEEE International Conference on Software Maintenance, pp. 274-283, 2009.

[26] A. Abran and P. Dominic, "Analysis of obvious outliers in ISBSG", on-going research, 2015.

[27] A. Abran ," work effort distribution across development phases", on-going, 2015.

[28] L. Hatton, “Reexamining the Fault Density-Component Size Connection,” IEEE Software, vol. 14, no. 2, pp. 89-97, April, 1997.

[29] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard P. W. Jones D. C. Hoaglin K. El Emam, and J. Rosenberg. “Preliminary guidelines for empirical research in software engineering”, IEEE Trans. Software Engineering, Vol. 28, pp. 721– 734, Aug 2002.

[30] F. Kerlinger. Foundations of Behavioral Research, Third Edition. Harcourt Brace Jovaonvich College Publishers, Orlando, Florida, 1986.

[31] N. Fenton, S. Pfleeger, Software Metrics: A Rigorous and Practical Approach, second ed., PWS Publishing Co., Boston, MA, 1999.

[32] SPSS Software, predictive analytics software and solutions, http://www-01.ibm.com/software/analytics/spss/

[33] A. J. Albrecht, "Measuring application development productivity." In Proceedings of the IBM Application Development Symposium, vol. 10, pp. 83-92. 1979.

[34] K. El Emam, S. Benlarbi, N. Goel, and S.N. Rai, “The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics,” IEEE Trans. Software Eng., vol. 27, no. 7, pp. 630-650, July 2001.