By no means : a study on aggregating software metrics

By No Means: A Study on Aggregating Software Metrics

Bogdan Vasilescu
Technische Universiteit Eindhoven
Den Dolech 2, P.O. Box 513, 5600 MB Eindhoven
The Netherlands
b.n.vasilescu@student.tue.nl

Alexander Serebrenik
Technische Universiteit Eindhoven
Den Dolech 2, P.O. Box 513, 5600 MB Eindhoven
The Netherlands
a.serebrenik@tue.nl

Mark van den Brand
Technische Universiteit Eindhoven
Den Dolech 2, P.O. Box 513, 5600 MB Eindhoven
The Netherlands
m.g.j.v.d.brand@tue.nl

ABSTRACT

Fault prediction models usually employ software metrics which were previously shown to be a strong predictor for defects, e.g., SLOC. However, metrics are usually defined on a micro-level (method, class, package), and should therefore be aggregated in order to provide insights in the evolution at the macro-level (system). In addition to traditional aggregation techniques such as the *mean*, *median*, or *sum*, recently econometric aggregation techniques, such as the Gini, Theil, and Hoover indices have been proposed. In this paper we wish to understand whether the aggregation technique influences the presence and strength of the relation between SLOC and defects. Our results indicate that correlation is not strong, and is influenced by the aggregation technique.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*corrections*; D.2.8 [**Software Engineering**]: Metrics—*complexity measures*

General Terms

Measurement, Economics, Experimentation

Keywords

Software metrics, maintainability, aggregation techniques

1. INTRODUCTION

Software maintenance is an area of software engineering with deep financial implications. Indeed, it was reported that up to 90% of the software budgets represent maintenance and evolution costs [10, 3]. Thus, in order to control software maintenance costs, it is desirable, e.g., to predict faulty components early in the development phase.

Fault prediction models usually employ software metrics which were previously shown to be a strong predictor for defects [9, 4, 21, 22, 20, 12]. Such a metric is size, measured in (source) lines of code, (S)LOC. Size (SLOC) not only corresponds to the intuitive belief that large systems have more faults in them than small systems, but was shown to act as an early indicator of problems better than, e.g., object-oriented metrics such as the Chidamber and Kemerer suite or the Lorenz and Kidd suite [9].

However, software metrics are commonly defined at micro-level (method, class, package), and should therefore be aggregated at macro-level (system), in order to provide insights in the study of maintainability and evolution.

Popular aggregation techniques include such standard summary statistical measures as *mean*, *median*, or *sum* [19]. Their main advantage is universality (metrics-independence): whatever metrics are considered, the measures should be calculated in the same way. However, as the distribution of many interesting software metrics is skewed [29], the interpretation of such measures becomes unreliable.

Alternatively, *distribution fitting* [6, 26, 29] consists of selecting a known family of distributions (e.g., log-normal or exponential) and fitting its parameters to approximate the metric values observed. The fitted parameters can be then considered as aggregating these values. However, the fitting process should be repeated whenever a new metric is being considered. Moreover, it is still a matter of controversy whether, e.g., software size is distributed log-normally [6] or double Pareto [14].

Recently, there is an emerging trend in using more advanced aggregation techniques, that are both reliable, as well as general. Examples of such approaches are the *Gini coefficient* [11], the *Theil index* [28], and the *Hoover index* [15], all well-known in econometrics for their applicability to studying income inequality [7], and recently applied to software metrics [27, 30, 13, 31].

In this preliminary study, based on the assumption that size is a good predictor for defects, hence size and defects should be statistically related, we wish to understand whether the aggregation technique influences the presence and strength of this relation. Briefly, our results indicate that correlation between SLOC and defects is not strong, and is influenced by the aggregation technique.

2. METHODOLOGY

We apply correlation analysis to SLOC data of Java classes aggregated at package level using different aggregation techniques, and defects (bug count per package). As a by-product of our evaluation, we also study the correlation between the different aggregation techniques themselves. The choice for aggregating data from class to package level rather

WETSoM'11, May 24, 2011, Waikiki, Honolulu, HI, USA

Table 1: Summary of the analyzed systems

	ArgoUML	Adempiere	Mogwai
Version	0.13.4	3.5.1a	2.6.0
#Java classes	1230	4047	2310
#Packages	94	152	365
#Bugs reported	89	303	143
#Bugs in SVN log	42	269	55
#Bugs mapped	39	163	38

than, e.g., from method to class level is motivated by the additional noise the latter would have introduced (while modifying a method in order to fix a bug, developers may touch a number of other methods, which are related to the method in question but not to the bug *per se*).

As case studies we have chosen three Java systems: ArgoUML, a popular UML modeling tool, Adempiere, an open-source ERP application, and Mogwai Java Tools, a Java Entity Relationship design and modeling (ERD) application. As aggregation techniques we have chosen the traditional *sum*, *mean*, and *median*, as well as the econometric inequality indices I_{Gini}, I_{Theil}, I_{Hoover}, I_{Kolm}, and I_{Atkinson} (see Section 3 for definitions and mathematical properties).

To study correlation between the aggregated metrics values and the number of bugs we started by choosing for each system the *version* with the highest number of bug fixes. The choice for bug *fixes* rather than *reports*, *dismissals* etc. follows [8] and is motivated by the fact that commit messages contain (at best) information only about the fixed bugs. This information is needed to map bugs to Java classes. Since we only analyze a snapshot of the case, the choice for the faultiest version ensures that the defect population is sufficiently large for the analysis to be accurate. Table 1 summarizes the three datasets of the study.

Next, the source code for each system was automatically processed and the list of classes contained in each package was built. We have considered packages containing at least 2 classes because the aggregation indices for packages containing one class only are equal to 0, hence should be excluded.

At the following step we mapped the defects to Java packages by analyzing the commit messages of the version control system log. Since the same class could have been affected multiple times during the fix of a known bug (e.g. because of a wrongly-implemented fix the first time), we only recorded it once in order to further minimize noise. Note the difference between the number of bugs reported in the bug tracker and the number of bugs mapped according to the version control system log. Apart from undocumented bug fixes, it is also due to some of the issues requiring changes to non-Java source files. The cardinality of the defect sets per package generated a list containing an element for each of the packages, and served as our validation metric.

Next, we calculated SLOC for each Java class and aggregated these values using the *mean*, *median*, *sum*, I_{Gini}, I_{Theil}, I_{Hoover}, I_{Kolm} and I_{Atkinson}.

Finally, we studied correlation between the aggregated values and defects, as well as between the aggregated values themselves. All computations were performed using R [25].

3. THEORETICAL COMPARISON

In this section we study a number of mathematical properties of the aggregation techniques to be empirically evaluated, relevant for their application to software metrics. We start by briefly presenting their mathematical definitions.

Let $\{x_1, \ldots, x_n\}$ be the collection of values to be aggregated. Then, the *sum*, denoted as x_{total}, is defined as $\sum_{i=1}^{n} x_i$. The *mean*, $\bar{x}$, is defined as $\frac{x_{total}}{n}$. The *median*, is defined as $x_{(n+1)/2}$ if n is odd, and $\frac{1}{2}(x_{n/2} + x_{n/2+1})$ if n is even. We further study the following econometric indices:

$$I_{\text{Gini}}(x_1, \ldots, x_n) = \frac{1}{2n\bar{x}} \sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j| \quad [18]$$

$$I_{\text{Theil}}(x_1, \ldots, x_n) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i}{\bar{x}} \log \frac{x_i}{\bar{x}} \right) \quad [28]$$

$$I_{\text{Hoover}}(x_1, \ldots, x_n) = \frac{1}{2} \sum_{i=1}^{n} \left| \frac{x_i}{x_{total}} - \frac{1}{n} \right| \quad [15]$$

$$I_{\text{Kolm}}(x_1, \ldots, x_n) = \log \left[\frac{1}{n} \sum_{i=1}^{n} e^{\bar{x} - x_i} \right] \quad [16]$$

$$I_{\text{Atkinson}}(x_1, \ldots, x_n) = 1 - \frac{1}{\bar{x}} \left(\frac{1}{n} \sum_{i=1}^{n} \sqrt{x_i} \right)^2 \quad [2],$$

where $|x_i - x_j|$ is the absolute value of $x_i - x_j$. In addition to I_{Theil} above, also known as the first Theil index, Theil [28] has also introduced the second Theil index, known as the mean logarithmic deviation. In this paper we do not consider the mean logarithmic deviation and whenever "the Theil index" is mentioned, I_{Theil} is meant. I_{Kolm} and I_{Atkinson} are standard instantiations of the Kolm and Atkinson families of indices, for parameters 1 and 0.5, respectively.

Domain.

Domain of the aggregation technique determines applicability of this technique to classes of software metrics. Econometric indices are usually applied to income or welfare distributions, i.e., to sets of positive values. Some software metrics, however, may have negative values, e.g., the maintainability index [23]. Since $\log z$ and $\sqrt{z}$ are undefined for $z < 0$, I_{Theil} and I_{Atkinson} are undefined as well. Unlike these indices, *mean*, *median*, *sum*, I_{Gini}, I_{Hoover}, and I_{Kolm} can be used to aggregate negative values. Moreover, as $\log 0$ is undefined, direct application of the Theil index formula is not possible. However, as shown in [27], I_{Theil} can be defined in presence of a zero value depending on whether zero denotes emptiness (e.g., SLOC) or not. Finally, formulas for the Gini index, the Theil index and the Atkinson index involve division by $\bar{x}$, while for Hoover index by x_{total}. Hence, they are undefined if $\bar{x} = 0$ and $x_{total} = 0$, respectively.

Since SLOC has non-negative values, all techniques here are appropriate for aggregating SLOC.

Interpretation.

Interpretation of the aggregated value depends on the range of the aggregation technique: e.g., 0.99 indicates a very high degree of inequality if I_{Gini} or I_{Hoover} is considered, while in case of I_{Theil} and I_{Atkinson} the interpretation would depend on the number of values being aggregated. The values obtained by applying the *mean*, *median*, or *sum* are unbounded. The Gini and the Hoover indices range over $[0, 1]$ if all the values being aggregated are positive. In general, this is not necessarily the case, e.g. $I_{\text{Gini}}(1, -1.5) = -2.5$ and $I_{\text{Hoover}}(1, -1.5) = 2.5$. Range of I_{Theil} and I_{Atkinson} depends on the number of values being aggregated: one can show that $0 \leq I_{\text{Theil}}(x_1, \ldots, x_n) \leq \log n$ and $0 \leq I_{\text{Atkinson}}(x_1, \ldots, x_n) \leq 1 - \frac{1}{n}$. The Kolm index ranges over non-negative reals.

Invariance.

We call the aggregation technique *invariant w.r.t. addition* if $I(x_1, \ldots, x_n) = I(x_1 + c, \ldots, x_n + c)$ for any $x_1, \ldots, x_n$

and c, provided $I(x_1+c,\ldots,x_n+c)$ exists. Similarly, we call the aggregation technique *invariant w.r.t. multiplication* if $I(x_1,\ldots,x_n) = I(x_1\cdot c,\ldots,x_n\cdot c)$ for any $x_1,\ldots,x_n$ and c, provided $I(x_1\cdot c,\ldots,x_n\cdot c)$ exists. Aggregating lines of code measured per file, aggregation-technique-invariant with respect to addition allows to ignore, e.g., headers containing the licensing information and included in all source files. Results obtained by applying an aggregation technique that is invariant with respect to multiplication are not affected if percentages of the total number of lines of code are considered rather than the number of lines of code themselves. The *mean* is neither invariant w.r.t. addition, nor to multiplication. It can be shown that I_{Gini}, I_{Theil}, I_{Hoover} and I_{Atkinson} are invariant with respect to multiplication. Unlike them, I_{Kolm} is invariant w.r.t. addition.

Decomposability.

Decomposability is the key property necessary for explanation of inequality by partitioning the values to be aggregated into disjoint groups. In econometrics such groups correspond, e.g., to education level, gender or ethnicity, while in software evolution research, e.g., to package, programming language and maintainer's name[27]. Formally, I is decomposable if for a partition $\{x_{1,1},\ldots,x_{1,n_1},\ldots,x_{J,1},\ldots,x_{J,n_J}\}$ of $\{x_1,\ldots,x_n\}$, $x_i \neq 0$, it holds that

$I(x_1,\ldots,x_n) = I(\bar{x}_1,\ldots,\bar{x}_J)+\sum_{j=1}^{J}(w_j\cdot I(x_{j,1},\ldots,x_{j,n_j}))$

for some coefficients $w_1,\ldots,w_J$ satisfying $\sum_{j=1}^{J} w_j = 1$, where $\bar{x}_j$ is the mean of $x_{j,1},\ldots,x_{j,n_j}$. If I is decomposable, then the ratio of the inequality between the groups and the total amount of inequality can be seen as the percentage of inequality that can be explained by partitioning the population into groups. Both I_{Theil} [7] and I_{Kolm} [17] are decomposable, while I_{Gini}, I_{Hoover}, and I_{Atkinson} are not [1]. While some authors propose decompositions of I_{Gini} or I_{Atkinson}, they use a different notion of decomposability [18].

4. RESULTS

To study correlation we have a choice between Kendall's $\tau$and the Pearson correlation coefficient r: while the latter requires normality of both distributions being compared, the former is applicable when the normality hypothesis can be rejected for at least one of the distributions. Thus, we conduct the Shapiro-Wilk normality test to determine the appropriate correlation statistics: for the defects vector the Shapiro-Wilk normality test allows to reject the normality hypothesis in all three cases (ArgoUML: $W = 0.80$, p-value $< 8.4\times10^{-5}$; Adempiere: $W = 0.24$, p-value $< 2.2\times10^{-16}$; Mogwai: $W = 0.36$, p-value $= 2.2\times10^{-16}$). Therefore, Kendall's τ should be used. Similar precautions were taken when studying the correlation between the different aggregation techniques themselves.

For correlation between SLOC and defects, the results are summarized in Table 2, where boldface corresponds to two-sided p-values not exceeding 0.01, and italics corresponds to those between 0.01 and 0.05. The following conclusions can be derived:

- Correlation with the number of defects always ranges from very low ($\tau \simeq 0.02$ for *mean* in ArgoUML) to medium ($\tau \simeq 0.51$ for *sum* in Adempiere). None of the techniques indicates strong and also statistically significant correlation with the number of defects.

Table 2: Correlation between results of different aggregation techniques and defects

	ArgoUML	Adempiere	Mogwai
mean	0.023	**0.392**	**0.197**
median	-0.142	**0.311**	**0.129**
sum	**0.313**	**0.510**	**0.151**
I_{Gini}	**0.267**	**0.225**	**0.134**
I_{Theil}	**0.269**	**0.185**	**0.135**
I_{Atkinson}	**0.245**	*0.168*	**0.138**
I_{Hoover}	**0.240**	0.113	*0.122*
I_{Kolm}	0.144	**0.412**	**0.204**

- Values aggregated using the *mean* indicate very inconsistent results. In ArgoUML *mean* shows very low correlation with defects, while in Mogwai *mean* together with I_{Kolm} indicate the strongest (among the techniques considered) and also statistically significant correlation with the number of defects.
- Values aggregated using the *sum* indicate the strongest (for ArgoUML and Adempiere) and second strongest (for Mogwai) correlation with the number of defects, which is also statistically significant. Although the correlation is not high, this confirms the intuition that large systems have more faults than small systems.
- Values aggregated using I_{Gini}, I_{Theil}, I_{Hoover}, and I_{Atkinson} indicate consistently similar correlation with the number of defects, although none of them ever indicates the strongest correlation. In fact, it turns out there is high and statistically significant correlation between aggregation techniques of this group, i.e., aggregation values obtained using these techniques convey the same information.

Threats to validity.

The results above should be considered preliminary and a number of threats to validity should be addressed in the future. With respect to construction validity we need to consider a more representative set of benchmarks and their versions. Furthermore, our information about the defects might be incomplete as not all defects might be recorded in the bug tracker, and our mapping of defects to classes might be imperfect due to limited recording of this information in the commit messages. Finally, we have considered only one metric, namely SLOC, and it is not clear whether the results obtained can be generalized to additional metrics.

5. CONCLUSIONS

In this paper we have presented the preliminary results of a study of the relation between size and defects, and the influence of the aggregation technique on this relation. We have discussed theoretical aspects of different aggregation techniques and applied them to aggregate lines of code values in ArgoUML, Adempiere, and Mogwai.

Our results suggest that correlation between SLOC and number of defects is not strong, which implies that size may not be a good predictor for defects as initially believed. However, the choice of aggregation technique does influence correlation of the aggregated values with the number of defects. We observed that values aggregated using the *mean* indicate very inconsistent correlation results, while values

aggregated using the *sum* indicate the strongest (for ArgoUML and Adempiere) and second strongest (for Mogwai) correlation with the number of defects, which is also statistically significant. I_{Gini}, I_{Theil}, I_{Hoover}, and I_{Atkinson} consistently indicate very high correlation among themselves. Although correlation between I_{Theil} and I_{Atkinson} can be explained by the close relation between the Atkinson family of inequality measures and Generalized Entropy measures (of which I_{Theil} is part), we have yet to understand their high correlation with I_{Gini} and I_{Hoover}.

A popular approach in the econometric literature consists of studying multiple econometric indices rather than focusing on one. For instance, [24] employs six different indices, including the Gini, Theil, and Atkinson indices studied here. Champernowne [5] has also observed that different indices exhibit different sensitivity to different "dimensions of inequality": while $1 - n^{I_{\text{Theil}}}$ was most sensitive to inequality associated with the exceptionally rich, I_{Gini} is second-most sensitive to inequality reflecting a wide spread of the less extreme incomes, without much tendency for the majority of them to be bunched within quite a narrow range.

Hence, as future work we consider identification of the dimensions of inequality most relevant for software metrics, and study of the most appropriate aggregation techniques. Furthermore, this theoretical investigation will be complemented by a more profound empirical research, similar to the preliminary study of Section 4, and including additional benchmark systems, and software and validation metrics. This study will also investigate the close relation between I_{Gini}, I_{Theil}, I_{Hoover}, and I_{Atkinson}. Finally, while in the current work only a single snapshot of each system has been considered, future work includes the study of differences between the econometric indices in the evolutionary settings.

6. REFERENCES

[1] S. Anand and S.M.R. Kanbur. The Kuznets process and the inequality–development relationship. *Journal of Development Economics*, 40(1):25–52, Feb. 1993.

[2] A.B. Atkinson. On the measurement of inequality. *Journal of Economic Theory*, 2(3):244–263, 1970.

[3] B.W. Boehm. *Software engineering economics*. Prentice Hall, 1981.

[4] L.C. Briand, J. .Wüst, J.W. Daly, and D.V. Porter. Exploring the relationship between design measures and software quality in object-oriented systems. *J. Syst. Softw.*, 51(3):245–273, 2000.

[5] D. G. Champernowne. A comparison of measures of inequality of income distribution. *The Economic Journal*, 84(336):787–816, 1974.

[6] G. Concas, M. Marchesi, S. Pinna, and N. Serra. Power-laws in a large object-oriented software system. *IEEE Trans. Software Eng.*, 33(10):687–708, 2007.

[7] F. A. Cowell. Measurement of inequality. *Handbook of Income Distribution*, 87–166. Elsevier, 2000.

[8] M. Eaddy, T. Zimmermann, K. D. Sherwood, V. Garg, G. C. Murphy, N. Nagappan, and A. V. Aho. Do crosscutting concerns cause defects? *IEEE Trans. Softw. Eng.*, 34:497–515, July 2008.

[9] 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. Softw. Eng.*, 27:630–650, 2001.

[10] L. Erlikh. Leveraging legacy system dollars for e-business. *IT Professional*, 2(3):17–23, 2000.

[11] C. Gini. Variabilitè e mutabilitè. *Studi Econornico-Giuridici della R. Univ. de Cagliari*, 1912.

[12] B. Goel, and Y. Singh. Empirical Investigation of Metrics for Fault Prediction on Object-Oriented Software. *Comp. Inf. Sci.*, 131:255-265, 2008.

[13] M. Goeminne, and T. Mens. Evidence for the Pareto principle in Open Source Software Activity. In *SQM*. CEUR-WS workshop proceedings, 2011.

[14] I. Herraiz. A statistical examination of the evolution and properties of libre software. In *ICSM*, pages 439–442. IEEE Computer Society, 2009.

[15] E.M. Hoover Jr. The measurement of industrial localization. *Rev. Eco. Stat.*, 18(4):162–171, 1936.

[16] S.-C. Kolm. Unequal inequalities I. *Journal of Economic Theory*, 12(3):416–442, 1976.

[17] F. A. Cowell and M.-P. Victoria-Feser. Robustness properties of inequality measures. *Econometrica*, 64(1):77–101, January 1996.

[18] P. J. Lambert and J. R. Aronson. Inequality decomposition analysis and the Gini coefficient revisited. *Economic Journal*, 103(420):1221–27, 1993.

[19] M. Lanza and R. Marinescu. *Object-Oriented Metrics in Practice*. Springer Verlag, 2006.

[20] R. Moser, W. Pedrycz, and G. Succi. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In *ICSE*, pages 181–190. IEEE, 2008.

[21] N. Nagappan, T. Ball, and A. Zeller. Mining metrics to predict component failures. In *ICSE*, pages 452–461. IEEE, 2006.

[22] H.M. Olague, L.H. Etzkorn, S. Gholston, and S. Quattlebaum. Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. *IEEE Trans. Software Engineering*, 33(6):402–419, 2007.

[23] P. Oman and J. Hagemeister. Construction and testing of polynomials predicting software maintainability. *Journal of Systems and Software*, 24(3):251–266, 1994.

[24] C. Papatheodorou and M. Petmesidou. Poverty profiles and trends: How do southern European countries compare to each other? In CROP int'l studies in poverty, 47–94. Zed Books, 2006.

[25] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010.

[26] A. Serebrenik, S. Roubtsov, and M.G.J. van den Brand. D_n-based architecture assessment of Java open source software systems. In *ICPC*, pages 198–207, IEEE Computer Society, 2009.

[27] A. Serebrenik and M.G.J. van den Brand. Theil index for aggregation of software metrics values. In *ICSM*, pages 1–9, IEEE Computer Society, 2010

[28] H. Theil. *Economics and Information Theory*. North-Holland, 1967.

[29] I. Turnu, G. Concas, M. Marchesi, S. Pinna, and R. Tonelli. A modified Yule process to model the evolution of some object-oriented system properties. *Inf. Sci.*, 181(4):883–902, 2011.

[30] R. Vasa, M. Lumpe, P. Branch, and O. Nierstrasz. Comparative analysis of evolving software systems using the Gini coefficient. In *ICSM*, pages 179–188, IEEE Computer Society, 2009.

[31] B. Vasilescu, and A. Serebrenik, and M.G.J. van den Brand. Comparative Study of Software Metrics' Aggregation Techniques. In *BeNeVol 2010*, Lille, France, pages 80–84, 2010.