Comparative study of software metrics' aggregation techniques

Comparative Study of Software Metrics' Aggregation Techniques

Bogdan Vasilescu, Alexander Serebrenik*, Mark van den Brand

Technische Universiteit Eindhoven,
Den Dolech 2, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Abstract

While software metrics are commonly used to assess software maintainability and study software evolution, they are usually defined on a micro-level (method, class, package). Metrics 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, recently econometric aggregation techniques such as the Gini index and the Theil index have been proposed. Advantages and disadvantages of different aggregation techniques have not been evaluated empirically so far. In this paper we present the preliminary results of the comparative study of different aggregation techniques.

Keywords:
software metrics, maintainability, aggregation techniques

1. Introduction

While software metrics are commonly used to assess software maintainability and study software evolution, they are usually defined on a micro-level (method, class, package). Metrics should therefore be aggregated in order to provide insights in the evolution at the macro-level (system). Popular aggregation techniques include the mean [14] and distribution fitting [4, 18]. The main advantage of the mean is its metrics-independence: whatever metrics are considered, the mean should be calculated in the same way. However, as the distribution of many interesting software metrics is skewed [22] the mean becomes unreliable. Distribution fitting consists of selecting a known family of distributions (e.g., log-normal, exponential or negative binomial) and fitting its parameters to approximate the metric values observed. 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 [4] or double Pareto [11].

It is highly desirable, hence, to develop an aggregation approach that would be both reliable and independent of the metrics being aggregated. Examples of such approaches are the Gini coefficient [10] and the Theil index [20], both well-known in econometrics [6] and recently applied to software metrics [21, 19]. Comparison of different aggregation techniques was so far missing, however. In this short paper we present the first preliminary results.

Remainder of this paper is organized as follows. In Section 2 we briefly introduce the aggregation techniques being compared. Section 3 compares the theoretical properties of different aggregation techniques. Section 4 described the empirical studies conducted and, finally, Section 5 discusses related work and concludes.

2. Aggregation techniques

In this section we briefly present the mathematical definitions of the aggregation techniques to be evaluated. Let $\{x_1, \ldots, x_n\}$ be the set of values to be aggregated. Then, *the mean*, denoted as $\bar{x}$, is defined as $\frac{1}{n}\sum_{i=1}^{n} x_i$.

*Corresponding author
Email addresses: b.n.vasilescu@student.tue.nl (Bogdan Vasilescu), a.serebrenik@tue.nl (Alexander Serebrenik), m.g.j.v.d.brand@tue.nl (Mark van den Brand)

The *Gini index* and the *Theil index* have already been applied to software metrics in [21, 19], respectively. In addition to these two econometric indices we also study the *Kolm index* and the *Atkinson index*:

$$I_{\text{Gini}}(x_1,\ldots,x_n) = \tfrac{1}{2n\bar{x}} \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j| \quad [13] \qquad I_{\text{Kolm}}(x_1,\ldots,x_n) = \log\left[\tfrac{1}{n}\textstyle\sum_{i=1}^{n} e^{\bar{x}-x_i}\right] \quad [12]$$

$$I_{\text{Theil}}(x_1,\ldots,x_n) = \tfrac{1}{n} \textstyle\sum_{i=1}^{n} \left(\tfrac{x_i}{\bar{x}} \log \tfrac{x_i}{\bar{x}}\right) \quad [20] \qquad I_{\text{Atkinson}}(x_1,\ldots,x_n) = 1 - \tfrac{1}{\bar{x}} \left(\tfrac{1}{n}\textstyle\sum_{i=1}^{n} \sqrt{x_i}\right)^2 \quad [2],$$

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 [20] 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 the standard instantiations of the Kolm and Atkinson families of indices, for a parameter values of 1 and 0.5, respectively.

3. Theoretical comparison

In this section we study a number of mathematical properties of the aggregation techniques relevant for their application to software metrics.

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 [15]. Since $\log z$ and $\sqrt{z}$ are undefined for $z < 0$, I_{Theil} and I_{Atkinson} are undefined as well. Unlike these indices, the mean, I_{Gini} and I_{Kolm} can be used to aggregate negative values. Moreover, as $\log 0$ is undefined direct application of the Theil index formula from Section 2 is not possible. However, as shown in [19] $I_{\text{Theil}}(x_1,\ldots,x_{n-1},x_n)$ can be defined for $x_n = 0$ depending on whether zero denotes emptiness (e.g., SLOC, number of classes in a package) or not. All other aggregation techniques considered in this paper can be applied to zero values. Finally, formulas for the Gini index, the Theil index and the Atkinson index involve division by $\bar{x}$. Hence, these indices are undefined if $\bar{x} = 0$. The mean and the Kolm index do not have additional cases when their values are undefined.

Range. 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} 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 can range from $-\infty$ to $+\infty$. The Gini index is often claimed to range over $[0, 1]$ [21]: this is, however, the case only if all the values being aggregated are positive. In general, this is not necessarily the case: $I_{\text{Gini}}(1, -1.5) = -2.5$. 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 say that the aggregation technique is *invariant with respect to 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 say that the aggregation technique is *invariant with respect to multiplication* 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. 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 with respect to addition nor to multiplication. It can be shown that I_{Gini}, I_{Theil} and I_{Atkinson} are invariant with respect to multiplication. Unlike these indices, I_{Kolm} is invariant with respect to 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[19]. Formally, I is decomposable if for any given partition $\{x_{1,1},\ldots,x_{1,n_1},\ldots,x_{J,1},\ldots,x_{J,n_J}\}$ of $\{x_1,\ldots,x_n\}$ it holds that

$$I(x_1,\ldots,x_n) = I(\bar{x_1},\ldots,\bar{x_J}) + \sum_{j=1}^{J}(w_j * I(x_{j,1},\ldots,x_{j,n_j}))$$

for some coefficients $w_1, \dots, w_J$ satisfying $\sum_{j=1}^{J} w_j = 1$, where $\bar{x}_j$ is the mean of $x_{j,1}, \dots, x_{j,n_j}$. 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} [6] and I_{Kolm} [5] are decomposable, while I_{Gini} and I_{Atkinson} are not [1]. It should be noted that while some authors propose means of decomposing I_{Gini} or I_{Atkinson}, they use a slightly different notion of decomposability [13, 7].

4. Empirical comparison

To perform empirical evaluation of different aggregation techniques we have conducted two series of experiments. As the case study we have chosen ArgoUML, a popular UML modeling tool written in Java.

In the first set of experiments we have applied correlation analysis to metrics data aggregated at package level using mean, I_{Gini}, I_{Theil}, I_{Kolm} and I_{Atkinson}, and defects (bug count per package). In the second set we have investigated the presence of correlation between the mean and the different indices, as well as between the different indices themselves, for the same metrics data.

The metric considered in this preliminary study is source lines of code (number of lines of code without comments and whitespace). The motivation for (S)LOC is twofold. First, previous research has showed that size, in terms of lines of code, is a strong predictor of defects [9]. Second, the same source mentions that although metrics such as the Chidamber and Kemerer suite or the Lorenz and Kidd suite were expected to be validated with respect to fault-proneness of classes (defects), after controlling for size none of the above metrics could be associated with defects anymore. Hence (S)LOC remains a reliable and easily-measurable predictor for defects.

4.1. Methodology

To study correlation between the aggregated metrics values and the number of bugs we have started by choosing the ArgoUML *version* with the highest number of bug fixes. The choice for bug *fixes* rather than *reports*, *dismissals* etc. is motivated by the fact that commit messages contain (at best) information only about the fixed bugs (typically indicated by keywords such as “issue” or “fix”). This information is needed in order to associate bugs with Java classes. Moreover, this follows the approach described in [8]. Since we only analyze a snapshot of the case, the choice for the faultiest version ensures that the defect population is sufficiently big to be accurate.

From the approximately 150 versions of ArgoUML released throughout its history, the version 0.13.4 has the highest number of bug fixes associated with it (89). It contains 94 packages and 1230 classes. Next, the source code of version 0.13.4 of ArgoUML was automatically processed and the *list of packages and Java classes* contained in each package was built. Next we have considered packages containing at least 2 classes: aggregation indices for packages containing one class only are equal to 0, and hence should be excluded. In total, 77 packages were considered.

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 minimize noise. Out of the 89 issues associated with version 0.13.4 of ArgoUML, there are only 41 mentioned in the commit log (e.g. because some of the issues required changes to non-Java source files). The cardinality of the defect sets generated a vector containing an element for each of the packages, and served as our validation metric.

Next we calculated SLOC for each Java class of the selected packages using CCCC[1], and aggregated these values using the mean, I_{Gini}, I_{Theil}, I_{Kolm} and I_{Atkinson}. Finally, in the first series of experiments we have studied correlation between the aggregated metrics vectors and the defects, while in the second series of experiments we have studied correlation between the aggregated metrics vectors themselves. All computations were performed using R [17].

4.2. Results

In the first series of experiments we have studied correlation between the aggregated metrics vectors and the defects. 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

[1]`http://cccc.sourceforge.net/`

Table 1: Correlation between results of different aggregation techniques and defects

	mean	I_{Gini}	I_{Theil}	I_{Kolm}	I_{Atkinson}	defects
mean		*0.170*	*0.192*	**0.676**	**0.203**	0.0096
I_{Gini}	*0.170*		**0.908**	**0.467**	**0.903**	**0.27**
I_{Theil}	*0.192*	**0.908**		**0.488**	**0.918**	**0.273**
I_{Kolm}	**0.676**	**0.467**	**0.488**		**0.501**	0.119
I_{Atkinson}	**0.203**	**0.903**	**0.918**	**0.501**		**0.229**

can be rejected for at least one of the distributions. Thus, we conduct the Shapiro-Wilk normality testto determine the appropriate correlation statistics: for the defects vector the Shapiro-Wilk normality test allows to reject the normality hypothesis (the W statistics equals 0.8003 and the p-value is 8.444×10^{-5}). Therefore, the Kendall's τ should be used. In the second series of experiments we have studied correlation between the values obtained for different aggregation techniques. Normality assumption can be rejected for the Theil, Kolm and Atkinson indices ($W_{\text{Theil}} = 0.8914$, $p_{\text{Theil}} = 7.68 \times 10^{-6}$; $W_{\text{Kolm}} = 0.6697$, $p_{\text{Kolm}} = 7.123 \times 10^{-12}$; and $W_{\text{Atkinson}} = 0.9248$, $p_{\text{Atkinson}} = 0.0002154$), so again the Kendall's τ should be used. Results of both studies are summarized in Table 1, where correlation results with two-sided p-values not exceeding 0.01 are typeset in boldface and those between 0.01 and 0.05 are typeset in italics.

Experiments seem to suggest that the aggregation techniques fall in two groups: one group consisting of I_{Gini}, I_{Theil} and I_{Atkinson}, another one of the mean and I_{Kolm}. There is high and statistically significant correlation between aggregation techniques of the same group, i.e., aggregation values obtained using these techniques convey the same information. Correlation between aggregation techniques of different groups ranges from low (0.17) to average (0.501) and is, in any case, lower than correlation between the results of the aggregation techniques of the same group. Most important, I_{Gini}, I_{Theil} and I_{Atkinson} indicate the strongest (among the techniques considered) and also statistically significant correlation with the number of defects. Among them, the highest τ value is obtained when the Theil index is used to aggregate the individual values ($\tau \simeq 0.273$) followed by the Gini index ($\tau \simeq 0.27$) and the Atkinson index ($\tau \simeq 0.229$). These results are statistically significant with two-sided p-values being 0.0014062, 0.0015996 and 0.0073827, respectively.

Although this evidence is preliminary, it is also important for several reasons. First, it provides an indication that the choice of aggregation technique leads to different correlation results with a validation set (in this case defects), even for simple software metrics such as lines of code. This finding prompts the need for additional research to determine if these relations are consistent both with respect to other software systems, as well as with respect to more complex metrics. Second, it corroborates the conjecture by which inequality indices such as the ones studied here can serve as viable alternatives to traditional aggregation methods such as the mean, when applied to software metrics. However, it is still a topic of research how using different aggregation techniques can affect the *interpretation* of a metric. For example, the fact that the inequality indices are equal to zero for packages with only one class seems to suggest that they shouldn't entirely replace the traditional aggregation techniques, but rather complement them. Similarly, the fact that high values for the inequality indices applied to some metrics (e.g. depth of inheritance tree), indicating high equality, may not always be desirable seems to suggest that special care is needed when using inequality indices in the evolutionary setting.

4.3. Threats to validity

The results presented above should be considered preliminary and a number of threats to validity should be addressed in the future. First of all, with respect to construction validity we need to consider a representative set of benchmarks rather than solely ArgoUML, and a representative set of their versions. Furthermore, our information about the defects might be incomplete as not all defects might be recorded in the issue tracking system, 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 lines of code, and it is not clear whether the results obtained can be generalized to additional metrics.

5. Conclusions

In this paper we have presented preliminary results of a comparative study of different aggregation techniques for software metrics. We have discussed theoretical aspects of different aggregation techniques and applied them to aggregate lines of code values in ArgoUML, version 0.13.4. Our results suggest that choice of the aggregation technique does influence correlation of the aggregated values with the number of defects, and that the Theil index, the Gini index and the Atkinson index lead to the highest correlation. Moreover, correlation between the values obtained for these aggregation techniques turned out to be very high.

Popular approach in the econometric literature consists in studying multiple econometric indices rather than focusing on one of them. For instance, [16] employs six different indices, including the Gini index, the Theil index and the Atkinson index studied in our paper. Champernowne [3] has indeed 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. As future work we, therefore, 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, software metrics and validation metrics. Finally, while in the current work only a single snapshot has been considered, the study of differences between the econometric indices in the evolutionary settings is also considered as future work.

Bibliography

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

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

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

[4] Giulio Concas, Michele Marchesi, Sandra Pinna, and Nicol Serra. Power-laws in a large object-oriented software system. *IEEE Trans. Software Eng.*, 33(10):687–708, 2007.

[5] Frank A. Cowell and Maria-Pia Victoria-Feser. Robustness properties of inequality measures. *Econometrica*, 64(1):77–101, January 1996.

[6] Frank Alan Cowell. Measurement of inequality. volume 1 of *Handbook of Income Distribution*, pages 87 – 166. Elsevier, 2000.

[7] Tarun Das and Ashok Parikh. Decomposition of inequality measures and a comparative analysis. *Empirical Economics*, 7:23–48, 1982. 10.1007/BF02506823.

[8] Marc Eaddy, Thomas Zimmermann, Kaitlin D. Sherwood, Vibhav Garg, Gail C. Murphy, Nachiappan Nagappan, and Alfred V. Aho. Do crosscutting concerns cause defects? *IEEE Trans. Softw. Eng.*, 34:497–515, July 2008.

[9] Kalhed El Emam, Saïda Benlarbi, Nishith Goel, and Shesh N. Rai. The confounding effect of class size on the validity of object-oriented metrics. *IEEE Trans. Softw. Eng.*, 27:630–650, 2001.

[10] Corrado Gini. Variabilitè e mutabilitè. *Studi Econornico-Giuridici della R. Universita de Cagliari*, 1912.

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

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

[13] Peter J. Lambert and J. Richard Aronson. Inequality decomposition analysis and the Gini coefficient revisited. *Economic Journal*, 103(420):1221–27, September 1993.

[14] Michele Lanza and Radu Marinescu. *Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems*. Springer Verlag, 2006.

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

[16] Christos Papatheodorou and Maria Petmesidou. Poverty profiles and trends: How do southern European countries compare to each other? In M. Petmesidou and C. Papatheodorou, editors, *Poverty and social deprivation in the Mediterranean: trends, policies, and welfare prospects in the new millennium*, CROP international studies in poverty research, pages 47–94. Zed Books, 2006.

[17] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010. ISBN 3-900051-07-0.

[18] Alexander Serebrenik, Serguei Roubtsov, and Mark van den Brand. D_n-based architecture assessment of Java open source software systems. In *Program Comprehension, 2009. ICPC '09. IEEE 17th International Conference on*, pages 198–207, May 2009.

[19] Alexander Serebrenik and Mark van den Brand. Theil index for aggregation of software metrics values. In *Software Maintenance, 2010. ICSM '10*, pages 1–9, September 2010.

[20] Henri Theil. *Economics and Information Theory*. North-Holland, 1967.

[21] Rajesh Vasa, Markus Lumpe, Philip Branch, and Oscar Nierstrasz. Comparative analysis of evolving software systems using the Gini coefficient. In *ICSM*, pages 179–188, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[22] Rajesh Vasa, Jean-Guy Schneider, and Oscar Nierstrasz. The inevitable stability of software change. In *ICSM*, pages 4–13. IEEE, 2007.