By no means: a study on aggregating software metrics

Vasilescu, B.N.; Serebrenik, A.; van den Brand, M.G.J.

Published in:

DOI:
10.1145/1985374.1985381

Published: 01/01/2011

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
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 byproduct 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
### Table 1: Summary of the analyzed systems

<table>
<thead>
<tr>
<th></th>
<th>ArgoUML</th>
<th>Adempiere</th>
<th>Mogwai</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>0.13.4</td>
<td>3.5.1a</td>
<td>2.6.0</td>
</tr>
<tr>
<td>#Java classes</td>
<td>1230</td>
<td>4047</td>
<td>2310</td>
</tr>
<tr>
<td>#Packages</td>
<td>94</td>
<td>152</td>
<td>365</td>
</tr>
<tr>
<td>#Bugs reported</td>
<td>89</td>
<td>303</td>
<td>143</td>
</tr>
<tr>
<td>#Bugs in SVN log</td>
<td>42</td>
<td>269</td>
<td>55</td>
</tr>
<tr>
<td>#Bugs mapped</td>
<td>39</td>
<td>163</td>
<td>38</td>
</tr>
</tbody>
</table>

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, $I_{\text{Gini}}$, $I_{\text{Theil}}$, $I_{\text{Hoover}}$, $I_{\text{Kolm}}$, and $I_{\text{Atkinson}}$.

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 $x_{\text{total}}$, is defined as $\sum_{i=1}^{n} x_i$. The mean, $\bar{x}$, is defined as $\frac{\text{sum}}{n}$. The median, $\hat{x}$, 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:

- $I_{\text{Gini}}(x_1, \ldots, x_n) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j|$ [18]
- $I_{\text{Theil}}(x_1, \ldots, x_n) = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{x_i}{\text{sum}} \log \frac{x_i}{\text{sum}} \right)$ [28]
- $I_{\text{Hoover}}(x_1, \ldots, x_n) = \frac{1}{n} \sum_{i=1}^{n} \frac{x_i}{\text{sum}} - \frac{\text{sum}}{n}$ [15]
- $I_{\text{Kolm}}(x_1, \ldots, x_n) = \log \left( \frac{1}{n} \sum_{i=1}^{n} \frac{x_i^2}{\text{sum}} \right)$ [16]
- $I_{\text{Atkinson}}(x_1, \ldots, x_n) = 1 - \frac{1}{\frac{n}{\text{sum}} \sum_{i=1}^{n} \sqrt{x_i}}$ [2],

where $|x_i - x_j|$ is the absolute value of $x_i - x_j$. In addition to $I_{\text{Theil}}$ above, also known as the first Theil index, $I_{\text{Theil}}$ [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, $I_{\text{Theil}}$ is meant. $I_{\text{Kolm}}$ and $I_{\text{Atkinson}}$ 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 $z < 0$, $I_{\text{Theil}}$ and $I_{\text{Atkinson}}$ are undefined as well. Unlike these indices, mean, median, sum, $I_{\text{Gini}}$, $I_{\text{Hoover}}$, and $I_{\text{Kolm}}$ 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], $I_{\text{Theil}}$ 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 $x_{\text{total}}$. Hence, they are undefined if $\bar{x} = 0$ and $x_{\text{total}} = 0$, 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 $I_{\text{Gini}}$ or $I_{\text{Hoover}}$ is considered, while in case of $I_{\text{Theil}}$ and $I_{\text{Atkinson}}$, 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 $I_{\text{Theil}}$ and $I_{\text{Atkinson}}$ 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 \( I_{\text{Gini}}, I_{\text{Theil}}, I_{\text{Hoover}} \) and \( I_{\text{Atkinson}} \) are invariant with respect to multiplication. Unlike them, \( I_{\text{Kolm}} \) 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

\[
I(x_1, \ldots, x_n) = I(\bar{x}_j) + \sum_{j=1}^{J} (w_i \cdot I(x_{i,1}, \ldots, x_{i,n_i}))
\]

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 \( I_{\text{Theil}} \) [7] and \( I_{\text{Kolm}} \) [17] are decomposable, while \( I_{\text{Gini}}, I_{\text{Hoover}}, \) and \( I_{\text{Atkinson}} \) are not [1]. While some authors propose decompositions of \( I_{\text{Gini}} \) or \( I_{\text{Atkinson}} \) 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 can be derived:

- 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

<table>
<thead>
<tr>
<th></th>
<th>ArgoUML</th>
<th>Adempiere</th>
<th>Mogwai</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>0.023</td>
<td>0.392</td>
<td>0.197</td>
</tr>
<tr>
<td>median</td>
<td>-0.142</td>
<td>0.311</td>
<td>0.129</td>
</tr>
<tr>
<td>sum</td>
<td>0.313</td>
<td>0.510</td>
<td>0.151</td>
</tr>
<tr>
<td>Gini</td>
<td>0.267</td>
<td>0.225</td>
<td>0.134</td>
</tr>
<tr>
<td>Theil</td>
<td>0.269</td>
<td>0.185</td>
<td>0.135</td>
</tr>
<tr>
<td>Atkinson</td>
<td>0.245</td>
<td>0.168</td>
<td>0.138</td>
</tr>
<tr>
<td>Hoover</td>
<td>0.240</td>
<td>0.113</td>
<td>0.122</td>
</tr>
<tr>
<td>Kolm</td>
<td>0.144</td>
<td>0.412</td>
<td>0.204</td>
</tr>
</tbody>
</table>
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. \( I_{\text{Gini}}, I_{\text{Theil}}, I_{\text{Hoover}}, \) and \( I_{\text{Atkinson}} \) consistently indicate very high correlation among themselves. Although correlation between \( I_{\text{Theil}} \) and \( I_{\text{Atkinson}} \) can be explained by the close relation between the Atkinson family of inequality measures and Generalized Entropy measures (of which \( I_{\text{Theil}} \) is part), we have yet to understand their high correlation with \( I_{\text{Gini}} \) and \( I_{\text{Hoover}} \).

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 \( 1 - n^{-1}I_{\text{Theil}} \) was most sensitive to inequality associated with the exceptionally rich, \( I_{\text{Gini}} \) 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 \( I_{\text{Gini}}, I_{\text{Theil}}, I_{\text{Hoover}}, \) and \( I_{\text{Atkinson}} \). 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


