Essay: A Metrics Suite for Object Oriented Design

Mats Stijlaart - University of Amsterdam, The Netherlands

November 29, 2015

This is an essay on the paper 'A Metrics Suite for Object Oriented Design' [4] by S. Chidamber and C. Kermerer which is dated from June 1994. First there will be a brief summary about the paper itself followed by its merits and drawbacks. The second paper used to write this essay is 'Quality Analysis of Object Oriented Cohesion Metrics' by P. Joshi and R. K. Joshi [10]. This paper gave insight to perform more research on the drawbacks of 'A Metrics Suite for Object Oriented Design'.

When object oriented design (OOD) starts to unfold in the industry, the need for feedback on this kind of software development increases. The writers address the problems that cause the need for a metrics suite that provides the required feedback on OOD. The existing metrics lack the theoretical ground, are not sufficient for OOD, technology dependent or too labor-intensive to collect. The paper introduces six metrics as a suite: Weighted Methods Per Class (WMC), Depth of Inheritance Tree (DIT), Number of Children (NOC), Coupling between object classes (CBO), Response For a Class (RFC) and Lack of Cohesion of Methods (LCOM). These metrics are assessed with the software complexity metric properties introduced by Weyuker [13] and found to comply with most of the specified properties. Criticism on previous research was the lack of theoretical basis, which is refuted on by the use of the model provided by Weyuker, and lack of empirical data, which is added by applying the metrics on two different OO applications (C++ and Smalltalk) using an automated tool. The main contribution of the paper can be summarized as a suite of six individual software complexity metrics that can be combined to illuminate design problems and/or choices within an OOD.

The paper is well structured and meets the reader's expectations. Where the writers introduce the problem (section II), they succeed this with separate sections on the point of view and approach. First the writers show how they approaches the properties introduced by OOD and which metrics were abstracted (section III), then they approach how the lack of theoretical ground is tackled in this research (section IV) and ends finally they provide a presentation of the empirical data collection (section V). Section VI contains the research results. The writers iterate over different metrics and accompany each metric with a point of view, perform an analytical evaluation and present the empirical data and its interpretation. This section concludes with a summary. The paper ends with the concluding remarks.

While the paper does not contain a research question the writers state a clear research problem, which is actually tackled. The problem is divided in two parts. The first is the lack of metrics that relate to the properties that emerge from an OOD, which indicate the need for such metrics. The second part concentrates on the theoretical criticism of metrics applied to conventional non-OO applications. Focusing on the second part, this is the argument to include a theoretical base in the research, but the provided substantiating research introduced by the writers merely focuses on complexity metrics instead of metrics in general. The writers lack to explicitly scope the research to complexity metrics, but abide to this point of view throughout the rest of the paper. With the constructive use of complexity as point of view and the use of a theoretical basis that reflected on the complexity measures by Weyuker, the writers should have addressed this scope in the title, abstract or introduction of the paper. Also the writers do not

introduce the concept of complexity. The concept of software complexity can be interpreted in different ways such as the difference of metrics introduced by McCabe [9] and Henry [6].

The writers argue, as mentioned above, that existing metrics do not relate to the properties of OOD, but they fail to demonstrate this or refer to previous work that indicates this. Although the reader can assume the lack of these metrics due to the maturity of object-oriented development in the industry at that time, the writers should have introduced their assumptions. The previous works that are referenced include suggestions and attempts for OOD metrics, but these sources do not address the missing relation of existiting metrics with the OOD properties, nor do the writers.

In the most important section of the paper the writers iterate over each proposed metric. The writers apply the same pattern for each metric, which implicitly sets the expectation for the reader. For each metric the writers give a clear definition, theoretical basis and point of view. The description and the theoretical basis give the definition and motivation for the metric. The point of view of each metric adds a more vivid argument for the need of each metric. Some of these arguments are more application-type specific. For example: "The larger the number of methods that can be invoked from a class, the greater the complexity of the class." I cannot reason about the types of application that were developed at the time of the paper, but in the present day there are a lot of APIs that separate their data from their logic. Although this is a more procedural approach than OO, it is applied with OO languages. The method count with these designs is higher in the data objects while they tend to be much simpler than the code in the domain-specific layer of an application. Aside from a few questionable point of views, the arguments the writers make are solid. While the point of view of the paper is convincing, the theoretical basis lacks detail on a few points. For example:

- The definition of the NOC metric states: "number of immediate subclasses subordinated of a class in the class hierarchy", but the theoretical basis states: "it is a measure of how many subclasses are going to inherit the methods of the parent class". The number of inheriting classes is not equal to the number of immediate subclasses of a class.

- The LCOM metric is a computation based on the intersection of accessed fields of two methods. To identify two methods with or without accessed field similarity, the writers use the notion of pairs. These pairs however are computed using the Cartesian product of a list containing field access sets (which are mapped from the methods). The Cartesian product holds the reflexive relational property, which by definition will decrease the outcome with at least the number of elements in the set consisting of the field access sets. The theoretical basis of LCOM states "the larger the number of similar methods, the more cohesive the class", but a simple counter example shows that there is a bug in the definition. Assume we have a set of instance variables $\{I_1, I_2\}$ and a set of methods $\{M_1, M_2\}$ where M_1 uses I_1 and M_2 uses I_2, then from the LCOM definition, $P = \{(\{I_1\}, \{I_2\}), (\{I_2\}, \{I_1\})\}$ and $Q = \{(\{I_1\}, \{I_1\}), (\{I_2\}, \{I_2\})\}$. This results in a LCOM value of 0 that indicates that the class is most cohesive, but M_1 and M_2 do not relate in a single matter. This problem is presented in other works such as Basisi et al. [12] and Chai et al. [3].

These points show that there is no sound symbiosis between the definition and the application.

After each metric is made clear the writers give an analytical evaluation of the metric and presents the empirical data. The analytical evaluation is a formal proof, which indicates that each metric complies to the metric properties specified by Weyuker.

The writers summarize the result in the last part of the paper, but also introduce new concepts on how the metrics combined could be used as design indicators, why some properties of the metrics model are not met and the reason for this, and the future direction to research the subject matter. I would rather have seen this in separate sections, because these components deserve their own section. The introduced points of view on combined metrics is one of the main contributions of the paper as it has a closer relation to OOD than the individual metrics.

The contribution of the paper to the field of software engineering is tremendous. A lot of researchers used the paper as basis for their research or simply as reference material. Mainly the cohesion metric has been researched a lot (possibly due to the demonstrated problem and researchers trying to fix this). In literature it is commonly known that there are different LCOM versions (1 up to 5) [2], which shows the extension on the initial contribution. Today's research tends to focus less on developing new metrics, but instead focuses more on how to apply the existing metrics. For example:

- the quality of cohesion metrics [1] [8],
- comparison between different cohesion metrics [10],
- and finding relations between metrics and faults [7] [11] [5].

In the present day there are no papers published with the form of 'A Metrics Suite for Object Oriented Design'. I expect that this has to do with the saturation on metric reseach and that it is simply not possible to publish a whole new suite of metrics. In contrast to this, most papers on metrics use one or more metrics introduced by Chidamber and Keremer, which shows the impact on the field.

To conclude, the paper is well structured and has a clear contribution. The research problem is clear. However, the approach tends to focus mainly on complexity metrics, the paper contains some aspects that could be described more explicitly, and the metric definitions have some inconsistencies. On the other hand the presented data is solid. As intended by the writers, the metrics can be used as indicators to prevent faults as Basili et al. [12] showed, and besides this the work is still used in modern research.

References

[1] Arm F. Desouky and Letha H. Etzkorn. "Object Oriented Cohesion Metrics: A Qualitative Empirical Analysis of Runtime Behavior". In: *Proceedings of the 2014 ACM Southeast Regional Conference*. 58. 2014.

[2] Linda Badri and Mourad Badri. "A Proposal of a New Class Cohesion Criterion: An Empirical Study". In: *Journal of Object Technology* 3.4 (2004), pp. 145–159.

[3] Heung Seok Chae. "A Cohesion Measure for Classes in Object-Oriented Systems". In: *Software Metrics Symposium*. 1996, pp. 158–166.

[4] Shyam R. Chidamber and Chris F. Kemerer. "A metrics suite for object oriented design". In: *IEEE Transactions on Software Engineering*. Vol. 20. 6. 1994, pp. 476–493. DOI: 10.1109/32.295895.

[5] Jehad Al Dallal. "Fault prediction and the discriminative powers of connectivity-based object-oriented class cohesion metrics". In: *Information and Software Technology*. Vol. 54. 2012, pp. 396–416.

[6] Sallie Henry and Dennis Kafura. "Software Structure Metrics Based on Information Flow". In: *IEEE Transactions on Software Engineering*. IEEE Computer Society, 1981, pp. 510–518.

[7] Hongmin Lu, Yuming Zhou, Baowen Xu, Harenton Leung and Lin Chen. "The ability of object-oriented metrics to predict change-proneness: a meta-analysis". In: *Empirical Software Engineering*. Vol. 17. 2012, pp. 200–242.

[8] Kalpana Johari and Arvinder Kaur. "Validation of Object Oriented Metrics Using Open Source Software System: An Empirical Study". In: *ACM SIGSOFT Software Engineering Notes*. Vol. 37. 2012, pp. 1–4.

[9] Thomas J. McCabe. "A Complexity Measure (Abstract)". In: *Proceedings of the Second International Conference on Software Engineering*. Ed. by Raymond T. Yeh and C. V. Ramamoorthy. IEEE Computer Society, 1976, p. 407.

[10] Padmaja Joshi and Rusikesh K. Joshi. "Quality Analysis of Object Oriented Cohesion Metrics". In: *Conference on the Quality of Information and Communications Technology*. 7. 2010, pp. 319–324.

[11] Pradeep Singh and Shrish Verma. "Empirical Investigation of Fault Prediction Capability of Object Oriented Metrics of Open Source Software ". In: *International Joint Conference on Computer Science and Software Engineering*. 2012, pp. 323–327.

[12] Victor R. Basili, Lionel C. Briand and Walcelio L. Melo. "A validation of object-oriented design metrics as quality indicators". In: *IEEE Transactions on Software Engineering*. IEEE Computer Society, 1996, pp. 751–761.

[13] Elaine J. Weyuker. "Evaluating software complexity measures". In: *IEEE Transactions on Software Engineering*. IEEE, 1988, pp. 1357–1365.