Toward Measuring Software Coupling via Weighted Dynamic Metrics

Henning Schnoor
Software Engineering Group, Kiel University
Kiel, Germany
henning.schnoor@email.uni-kiel.de

Wilhelm Hasselbring
Software Engineering Group, Kiel University
Kiel, Germany
hasselbring@email.uni-kiel.de

ABSTRACT

Coupling metrics are an established way to measure internal software quality with respect to modularity. Dynamic metrics have been used to improve the accuracy of static metrics for object-oriented software. We introduce a dynamic metric *NOI* that takes into account the number of interactions (method calls) during the run of a system. We used the data collected from an experiment to compute our NOI metric and compared the results to a static coupling analysis. We observed an unexpected level of correlation and significant differences between class- and package-level analyses.

ACM Reference Format:
Henning Schnoor and Wilhelm Hasselbring. 2018. Toward Measuring Software Coupling via Weighted Dynamic Metrics. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3195000

1 INTRODUCTION

Coupling [11]—the number of inter-module interactions in software systems—has long been identified as a software quality metric for modularity [10]. High cohesion and low coupling is a design guideline in software engineering. For microservice architectures, low coupling among microservices is of particular relevance [8, 9].

The coupling degree of a module (class or package) is usually measured statically, based on source or compiled code. For object-oriented software, static coupling measurement often fails to account for effects of inheritance with polymorphism and dynamic binding [3]. Dynamic analysis addresses these issues, using monitoring logs generated during the run of the software. Usually, such analyses use the data to detect the occurrence of methods calls, but do not take the frequency of these calls into account.

We use dynamic analysis for *weighted* measurements and count the *number of interactions*, NOI, during program execution. The *NOI coupling degree* of a module **A** is the number of method calls of **A**. This results in three flavors of the metric, considering import, export, or combined coupling. NOI and static coupling degrees are very different: A class with low static coupling degree does not necessarily have low NOI coupling degree, since a method call appearing once in the static analysis can be performed millions of times during runtime. Our analysis investigates the relationship between our dynamic NOI metric and static coupling measures.

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195000

Contributions

We performed an experiment monitoring usage of Atlassian JIRA [4] over four weeks, and computed our NOI metric based on the obtained data. We compared these results to static coupling degrees. Our results show that NOI and static coupling are different but correlated. When considering package-level coupling, the correlation is significantly stronger than for class-level coupling. A possible interpretation of this result is that effects like polymorphism and dynamic binding often do not cross package boundaries.

Related Work

Allier et al. [1] compare static and dynamic metrics. Dynamic (unweighted) metrics have been investigated in numerous papers (see, e.g., Arisholm et al. [3] as a starting point, also the surveys by Chhabra and Gupta [6] and Geetika and Singh [7]). Yacoub et al. [13] use weighted metrics. However, to obtain the data, they do not use runtime instrumentation but "early-stage executable models." They also assume a fixed number of objects during the software's runtime, while our approach also allows dynamic object instantiation.

2 INITIAL EXPERIMENTS

Our dynamic analysis is based on an experiment monitoring Atlassian JIRA, version 7.3.0 [4] using the Kieker monitoring framework [12] for dynamic analysis and Apache BCEL (Byte Code Engineering Library) [2] for static analysis. The workload used for the dynamic analysis contained 196,442,043 logged method calls, the maximal throughput was 176,116 monitored actions per second.

Preliminary Results

A central goal of this paper is to study the relationship between static and dynamic couplings. We compare two fundamentally different ways to measure coupling of modules:

(1) The static analysis counts, for each module **A**, the number of modules **B** to which **A** is connected via a method call.
(2) Our dynamic analysis computes the NOI metric: For each module **A**, it counts the number of interactions to any module **B** during a given run of the system.

We first studied the distribution of coupling degrees in our two types of analyses. For export coupling on a class level, the two analyses lead to similar results (see Figures 1 and 2). The difference between import coupling degrees was significantly higher. In the figures, for the ratios $\alpha_1 = 0.2\%$, $\alpha_2 = 0.4\%$, …, $\alpha_k = 100\%$, the bar above the value α_i indicates the ratio of modules whose coupling

degree is between α_{i-1} (or 0 if $i = 1$) and α_i (where 100% is the maximal occurring degree). The dashed line indicates the position of the mean coupling degree on the x-axis.

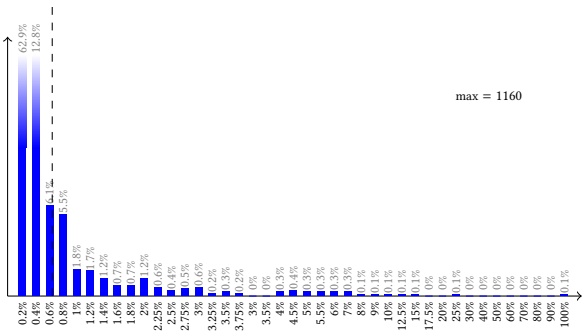

Figure 1: Static coupling degree for classes

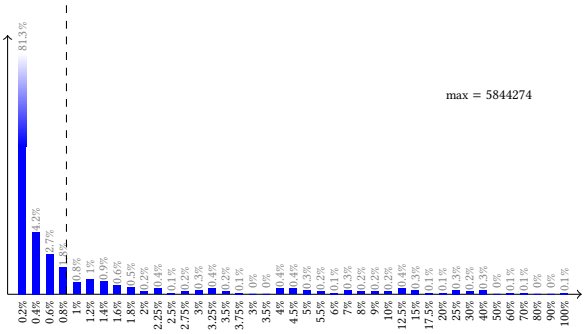

Figure 2: Import NOI for classes

As a second analysis, we compared the coupling degrees of individual modules. Obviously, the raw numerical values cannot be compared meaningfully. However, a key reason to use metrics is to identify modules with the highest coupling degrees. Therefore, we analyze the relationship between the *ranks* among the modules in the two analyses: Each analysis yields a *coupling-rank* of the modules, ranking the ones with highest coupling degree first. We compare these ranks using the Kendall-Tau distance [5]. Values smaller than 0.5 can be interpreted as the ranks being closer to each other than expected from two random ranks. Values larger than 0.5 indicate the opposite. Values further away from 0.5 imply higher correlation. Depending on whether we aggregate at the class- or package-level and on whether we consider import, export, or combined coupling, we obtain the following six distance values:

	class-level	package-level
import	0.45	0.36
export	0.41	0.33
combined	0.41	0.33

Discussion

All distances are below 0.5, many of them significantly so. This, and the coupling degree distribution, indicate a significant similarity between our NOI metric and static coupling measures and suggests that these different types of analyses are correlated. As argued above, this was not necessarily to be expected. Hence, the NOI coupling degree seems to give additional, but not unrelated information compared to the static case.

In all three cases, the distance in the package case is smaller than the distance in the class case, sometimes significantly. A possible explanation is that in the package case, the object-oriented effects that are often cited as the main reasons for performing dynamic analysis are less present, as, e.g., inheritance relationships are often between classes residing in the same package.

3 CONCLUSIONS

We compared static and NOI measurements. Our preliminary results suggest that dynamic coupling metrics complement their static counterparts: Despite the large (and expected) difference, there is significant correlation. This suggests that further study of dynamic quantitative measurements is an promising line of research. In particular, it will be interesting to investigate how these metrics can be used to evaluate the quality of software systems.

REFERENCES

[1] Simon Allier, Stéphane Vaucher, Bruno Dufour, and Houari A. Sahraoui. 2010. Deriving Coupling Metrics from Call Graphs. In *Tenth IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2010, Timisoara, Romania, 12-13 September 2010*. IEEE Computer Society, 43–52. https://doi.org/10.1109/SCAM.2010.25

[2] Apache Software Foundation. 2016. Commons BCEL: Byte Code Engineering Library. (2016). https://commons.apache.org/proper/commons-bcel/.

[3] Erik Arisholm, Lionel C. Briand, and Audun Føyen. 2004. Dynamic Coupling Measurement for Object-Oriented Software. *IEEE Trans. Software Eng.* 30, 8 (2004), 491–506. https://doi.org/10.1109/TSE.2004.41

[4] Atlassian. 2018. JIRA Project and issue tracking. (2018). https://www.atlassian.com/software/jira/.

[5] Lionel Briand, Khaled El Emam, and Sandro Morasca. 1996. On the application of measurement theory in software engineering. *Empirical Software Engineering* 1, 1 (01 Jan 1996), 61–88. https://doi.org/10.1007/BF00125812

[6] Jitender Kumar Chhabra and Varun Gupta. 2010. A Survey of Dynamic Software Metrics. *J. Comput. Sci. Technol.* 25, 5 (2010), 1016–1029. https://doi.org/10.1007/s11390-010-9384-3

[7] Rani Geetika and Paramvir Singh. 2014. Dynamic Coupling Metrics for Object Oriented Software Systems: A Survey. *SIGSOFT Softw. Eng. Notes* 39, 2 (March 2014), 1–8. https://doi.org/10.1145/2579281.2579296

[8] Wilhelm Hasselbring. 2016. Microservices for Scalability: Keynote Talk Abstract. In *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering (ICPE 2016)*. ACM, New York, NY, USA, 133–134. https://doi.org/10.1145/2851553.2858659

[9] Wilhelm Hasselbring and Guido Steinacker. 2017. Microservice Architectures for Scalability, Agility and Reliability in E-Commerce. In *Proceedings 2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. IEEE, Gothenburg, Sweden, 243–246. https://doi.org/10.1109/ICSAW.2017.11

[10] David Lorge Parnas. 1972. On the Criteria to Be Used in Decomposing Systems into Modules. 15, 12 (Dec. 1972), 1053–1058. https://doi.org/10.1145/361598.361623

[11] W. Stevens, G. Myers, and L. Constantine. 1979. Structured Design. In *Classics in Software Engineering*, Edward Nash Yourdon (Ed.). Yourdon Press, Upper Saddle River, NJ, USA, 205–232. http://dl.acm.org/citation.cfm?id=1241515.1241533

[12] André van Hoorn, Jan Waller, and Wilhelm Hasselbring. 2012. Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE 2012)*. ACM, 247–248. https://doi.org/10.1145/2188286.2188326

[13] Sherif M. Yacoub, Hany H. Ammar, and Tom Robinson. 1999. Dynamic Metrics for Object Oriented Designs. In *6th IEEE International Software Metrics Symposium (METRICS 1999), 4-6 November 1999, Boca Raton, FL, USA*. IEEE Computer Society, 50–61. https://doi.org/10.1109/METRIC.1999.809725