Calibrating program slicing metrics for practical use

David Bowes
Hertfordshire University
d.h.bowes@herts.ac.uk

Steve Counsell
Brunel University
steve.counsell@brunel.ac.uk

Tracy Hall
Brunel University
tracy.hall@brunel.ac.uk

Abstract
Program slicing metrics are an important addition to the range of static code measures available to software developers and researchers alike. However, slicing metrics still remain under-utilized due partly to the difficulty in calibrating such metrics for practical use; previous use of slicing metrics reveals a variety of calibration approaches. This paper reports on the effect of including different variables in the calibration (*and collection*) *of slicing metrics. Findings suggest a variety of different results depending on the level of abstraction at which the metrics are collected (system or file), the inclusion or exclusion of 'printf' statements, formal 'ins', 'outs' and global variables in the metrics' calculation..*

1. Introduction

Weiser [12, 13] and Ott and Thuss [9] defined a set of slice based metrics including Tightness, Coverage, Overlap, Min. Coverage and Max Coverage (see Table 1). These metrics could, potentially, be an important addition to the current range of static code metrics used by developers. Although slicing metrics have been studied in relation to code cohesion [7, 8, 9] they are rarely used by developers. Previous studies exploring the efficacy of slice-based metrics [2, 3, 4, 5, 6, 7, 8, 9, 10, 11] have tended to use different sets of variables in specifying the slices on which the metrics are based. This wide variety of approaches to data collection has made it difficult for developers (and indeed researchers) to interpret and subsequently use slice-based metrics. The aim of this paper is to provide a standard baseline upon which future use and research into slice-based metrics can be based.

2. Results

We investigated how each of the four categories of principal variables shown in Table 2 contributes empirically to the slice-based metrics of: Tightness, Overlap, Coverage, Min Coverage and Max Coverage for the Barcode Open Source Project. The results are shown in Table 3, from which a number of features emerge. First, inter-module effects are reduced significantly by slicing files individually compared to slicing on a whole project. The values on the right-hand side of Table 3 (Figure 2) are generally smaller than those on the left-hand side (Figure 1). The most notable reduction is for the Tightness metric, whose computation is heavily influenced by the cardinality of slice intersections.

Table 1. Weiser's slice-based metrics

Metric	Formula
Tightness	$= \frac{\lvert SL_{\text{int}} \rvert}{length(M)}$
Overlap	$= \frac{1}{\lvert V_o \rvert} \sum_{i=1}^{\lvert V_o \rvert} \frac{\lvert SL_{\text{int}} \rvert}{\lvert SL_i \rvert}$
Coverage	$= \frac{1}{\lvert V_o \rvert} \sum_{i=1}^{\lvert V_o \rvert} \frac{\lvert SL_i \rvert}{length(M)}$
Min Coverage	$= \frac{\min_i \lvert SL_i \rvert}{length(M)}$
Max Coverage	$= \frac{\max_i \lvert SL_i \rvert}{length(M)}$

Key	
Line	Line of code
M	Set of program lines in a module NB $length(M) \equiv \lvert M \rvert$
V_0	Set of variables used to slice a module
SL_i	Set of program lines in the slice for the *i*'th variable in V_0
SL_{int}	Intersection of all slices formed from each V_0

(Tightness measures the proportion of the module which is common to all slices; Overlap is the average proportion of common slices compared to each slice; Coverage is the average length of each slice compared to the length of the module; Min Coverage is ratio of the shortest slice compared to the length of the module and Max Coverage is the ratio of the longest slice compared to the length of the module; V_o is the set of output variables)

Table 2. Categories of principal variables and their use frequency (#) in previous studies

Categories	Description	# Studies
Formal ins	Input parameters for the function specified in the module declaration	6

Formal outs	The set of return variables	8
Global variables	The set of variables which are used or may be affected by the module	9
Printfs	Variables which appear as formal outs in the list of parameters in an output statement (e.g. printf)	7

NB: 'Module' is a function or method *not* a set of functions or files acting together.

Table 3. Average module metrics for different combinations of variables

Variables				Average module metrics									
				Sliced as a project					Files sliced individually				
I	O	G	pF	Over-lap	Tight-ness	Cover-age	Min C	Max C	Over-lap	Tight-ness	Cover-age	Min C	Max C
✓	✓	✓	✓	0.859	0.814	0.919	0.828	0.984	0.649	0.481	0.691	0.523	0.901
✓	✓	✓		0.861	0.820	0.926	0.833	0.984	0.643	0.482	0.705	0.524	0.901
✓	✓		✓	0.903	0.857	0.917	0.870	0.984	0.712	0.551	0.717	0.588	0.898
✓		✓	✓	0.905	0.852	0.926	0.863	0.977	0.759	0.563	0.712	0.587	0.892
	✓	✓	✓	0.898	0.837	0.918	0.842	0.966	0.745	0.519	0.671	0.543	0.845
✓	✓			0.911	0.869	0.929	0.881	0.984	0.728	0.560	0.743	0.590	0.898
		✓	✓	0.891	0.840	0.927	0.852	0.981	0.772	0.518	0.653	0.538	0.820
✓			✓	0.947	0.895	0.928	0.905	0.975	0.839	0.672	0.764	0.694	0.885
	✓	✓		0.920	0.844	0.915	0.847	0.953	0.767	0.521	0.653	0.544	0.761
✓		✓		0.911	0.869	0.929	0.881	0.984	0.728	0.560	0.743	0.590	0.898
	✓		✓	0.949	0.883	0.914	0.886	0.956	0.820	0.591	0.688	0.610	0.792
✓				0.972	0.929	0.951	0.933	0.975	0.944	0.823	0.856	0.832	0.885
	✓			1.000	0.897	0.897	0.897	0.897	1.000	0.612	0.612	0.612	0.612
		✓		0.907	0.859	0.941	0.866	0.971	0.851	0.538	0.639	0.547	0.717
			✓	0.917	0.851	0.896	0.866	0.968	0.749	0.464	0.597	0.496	0.778

I = Formal Ins, O = Formal Out, G = Globals, pF=printf; NB: Both forward and backward slices were used in all cases.

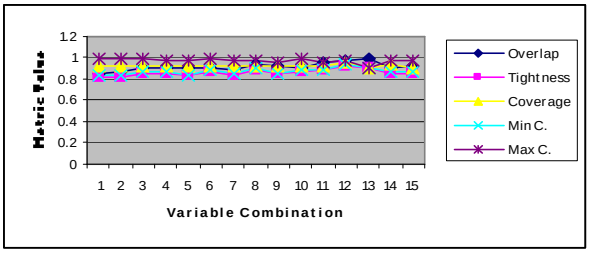

Figure 1. Sliced as a project

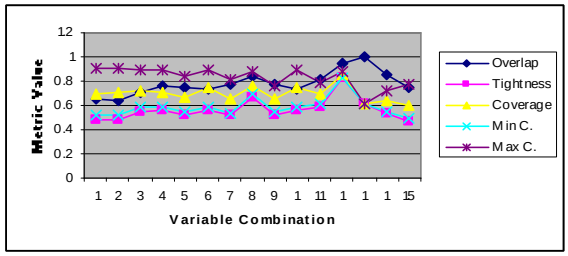

Figure 2. Files sliced individually

The least sensitive metrics appear to be those for Max Coverage, a metric influenced only by the size of the maximum slice. Second, exclusion of global variables (G) seems to cause a rise in the values of many of the metrics, both on the left *and* right-hand side of Table 3. This may be explained by the need to consider extra module inter-relationships induced by the use of global variables (which tend to push down the values of the metrics). The same effect can not be said of the inclusion or exclusion of printfs or formal ins (I) and outs (O), none of which seem to significantly influence the values of the metrics.

3. Acknowledgements

This research is kindly supported by grant from the EPSRC of the UK (EP/E055141/1).

References

[1] S. Counsell and D Bowes "A Theoretical and Empirical Analysis of Slicing Metrics for Cohesion Based on Matrices", submitted to CSMR 2008.
[2] J. Bieman, L. Ott. "Measuring functional cohesion." IEEE Trans. on Soft Eng., 20(8)::644–657, 1994.
[3] M. Harman, S. Danicic, B. Sivagurunathan, B. Jones, and Y. Sivagurunathan. "Cohesion metrics." International Software Quality Week, San Francisco, CA, USA, 1995.
[4] M. Harman. "Cleaving together - program cohesion with slices." EXE., Vol. 11 Iss. 8 pp. 35–42, 1997.
[5] A. Lakhotia. "Rule-based approach to computing module cohesion." International Conf. on Software Engineering, Baltimore, USA, 1993. pp. 35–44.
[6] T. M. Meyers and D. Binkley. "A longitudinal and comparative study of slice-based metrics." International Software Metrics Symposium, Chicago, USA, 2004.
[7] T. M. Meyers and D. Binkley. "Slice-based cohesion metrics and software intervention." Working Conference on Reverse Engineering, pp. 256–265, 2004.
[8] T. M. Meyers and D. Binkley. "An empirical study of slice based cohesion and coupling metrics." ACM Trans. on Software Engineering and Methodology, 2007.
[9] L. Ott and J. Thuss. "The relationship between slices and module cohesion." International Conference on Software Engineering, pp. 198–204, 1989.
[10] K. Pan, S. Kim, and E. J. Whitehead Jr. "Bug classification using program slicing metrics." IEEE Workshop on Source Code Anal and Manip., 31–42, 2006.
[11] M. Weiser. Program Slices: Formal, psychological, and practical investigations of an automatic program abstraction method. PhD Thes., Univ. Mich., MI, 1979.
[12] M. Weiser. "Program Slicing" Intl. Conference on Software Eng., San Diego, USA, 1981. pp. 439–449.
[13] Grammatech Inc. The CodeSurfer slicing system, 2000.