Synthesized dataset for search-based test data generation methods focused on MC/DC criterion

Ján Čegiň
Institute of Informatics, Information Systems and Software Engineering Slovak University of Technology in Bratislava
Bratislava, Slovakia
jan.cegin@stuba.sk

Karol Rástočný
Institute of Informatics, Information Systems and Software Engineering Slovak University of Technology in Bratislava
Bratislava, Slovakia
karol.rastocny@stuba.sk

Mária Bieliková
Slovak Research Center for Artificial Intelligence slovak.AI
Bratislava, Slovakia
maria.bielikova@slovak.ai

***Abstract*—Unit testing focused on the Modified Condition/Decision Coverage (MC/DC) criterion is essential in development of safety-critical systems as recommended by international standards. Designing unit tests for such specific software is time-consuming task which can be partially automated by test data generation methods. Special attention is given to search-based methods which are often used for problems where traditional methods like symbolic execution fall short. However, no publicly available dataset for evaluation of such methods taking into account specifics of the MC/DC criterion, which is esential for safety-critical systems. In this paper we present an analysis of software of safety-critical systems and we postulate to find a fitting open source project which could serve as a synthesized dataset for future evaluations of search-based test data generation methods for the MC/DC criterion.**

***Index Terms*—MC/DC criterion, unit testing, search-based testing, test data generation, safety-critical systems**

I. Introduction

Safety-critical systems are nowadays present in multiple domains and used in various parts of everyday life. Aviation, automotive, healthcare and many more are some of the domains where safety-critical systems are present. Software of these systems is tested via multiple techniques, which are recommended for use in standards such as IEC 61508 [1] (general standard for safety critical systems) and ISO 26262 [3] (automotive industry). Both international standards recommend the Modified Condition/Decision Coverage (MC/DC) criterion for designing unit tests [2]. MC/DC is a very strong criterion which greatly reduces the number of tests needed to meet necessary condition coverage, whilst preserving necessary quality attributes.

There have been proposed methods to generate test data that conform with the MC/DC criterion [4]. Test data generation is a very important and difficult field of software engineering. It is because its results affect the cost of software development as well as the effectiveness of testing. Search-based methods look at the problem of generating test data as a computational search problem. These methods are often used for problems, where traditional test data generation techniques [4] (symbolic execution, model based, etc.) fail short. But problems exist with the evaluation of such methods, as no public dataset is present for complex evaluation of different techniques.

In this paper, we address a problem of missing public dataset for evaluation of test data generation methods conforming with the MC/DC criterion. We present an analysis of current evaluation techniques used in this domain. Secondly, we present a first analysis of software of an automotive safety-critical system and we postulate properties of needed synthesized dataset based on the analysis of the real world industrial project.

II. Evaluating search-based test data generation methods for MC/DC criterion

Some of the search-based methods [2], [5] for the MC/DC criterion use hand-picked classes and methods that were previously used in works they are comparing with. Most methods use the Triangle classification algorithm — a program which classifies a triangle base on edge sizes. However, this algorithm differs significantly in its implementations, as pointed out in the method [5] which also evaluated itself on four different implementations of this algorithm. This leads to varying complexity of this algorithm used in evaluations.

Other methods [8], [9] evaluate themselves on multiple benchmarks, most notably programs that have been used previously with the inclusion of some new open-source methods they picked. Methods are often picked to contain known defects to evaluate, if the given method does not only improve coverage, but if it also finds these known defects.

There are also methods [6], [7] that were evaluated on industrial programs or even complete industrial projects. However, these projects are not available for research, as these projects are often legal bounded to not be published. Published metrics from these projects remain at a basic level (number of classes, lines of code, etc.).

As such, the current evaluations of search-based methods for test data generation for branch or MC/DC criteria are of varying quality, ranging from only one method used, to using multiple industrial projects. This makes replicating results as well as comparing methods difficult.

III. Synthesised dataset construction

We present our work on a synthesised dataset based upon real industrial project which is an embedded software system for control of car parts from our industrial partner. This system fails under the SIL 4 level defined in the IEC 61508 [1]. A synthesized dataset for evaluation of test data generation for combinatorial testing has been created and used [10] before. We propose similar approach for synthesizing a dataset for search-based methods and MC/DC criterion.

We propose the following process for the synthesised dataset construction. Firstly, we collect metrics for available safety-critical systems, present and analyze them as so we can provide a comparison using software metrics with open-source projects written in the same language. This give us an insight what features differentiate software of safety-critical systems from open-source software, if some do at all. Using this knowledge, we can find a suitable open-source projects which would be close to the analyzed safety-critical systems using the collected metrics. Such projects can be processed by various test data generation methods. Finally, we select functions, generated unit tests and achieved coverage, which are similar to the ones present in the analyzed safety-critical systems. This provides a dataset for evaluation of future test data generation methods from the provided baseline, as well as a reasonable comparison of different methods and their applicability to unit testing for the MC/DC criterion.

From our preliminary results, we present software metrics analysis of the software developed by our industrial partner. These metrics were collected per function, as the focus is on unit testing, with more than 1100 functions. We selected the most commonly used software metrics such as Logical Source Lines of Code (LSLOC) and McCabe Cyclomatic complexity. Our findings are summarized in Table I. To further emphasize the possible specifics that software of safety critical systems might have, we also propose these metrics:

- *Condition complexity (CCom)*: number of *and* and *or* logical operations used in conditions inside analyzed function. We propose this metric to measure condition complexity which directly affects the number of needed test cases for MC/DC criterion.
- *Function calls (FAll)*: number of all function calls inside the function.
- *Function calls (Funq)*: number of unique function calls inside the function. We propose function call–based metrics to emphasize the number of external function calls inside the analyzed function.
- *Bitwise operations (NBit)*: number of bitwise operations used in the function. We propose this metric as bitwise operations used for memory manipulation are common in safety-critical systems.

It should be also noted that the IEC 61508 [1] forbids such code constructs as recursion and memory manipulation, as it only allows static memory on the SIL 4 level. As such, we search for open-source projects that avoid such code constructs, or keep their usage at a minimal level.

TABLE I
Metrics per function for the analysed software

Metric	Avg	Std	25%	50%	75%	Min	Max
LSLOC	16	15	7	12	20	2	124
McCabe	7.6	7.5	3	5	9	2	53
CCom	0.8	1.4	0	0	1	0	12
FAll	3.2	5.4	0	1	4	0	54
Funq	1.6	3	0	0	2	0	32
Nbit	4.8	10	0	0	5	0	85

IV. Conclusion and future work

We address the problem of missing publicly available dataset for evaluation of test data generation methods for unit testing conforming with the MC/DC criterion. We assume that there exist differences between open-source software projects and software of safety-critical systems. Based on this we analyzed real automotive safety-critical system. Using this analysis, we will outline differences between open-source software and the domain specific software of safety-critical systems. This analysis will serve for fitting open-source projects, which would have at least similar features.

Next, we plan to expand the analysis to open-source projects and to compare results of analysed projects. This should provide us with information as to which projects are fitting for a dataset synthesis. Then we plan to produce unit tests conforming with MC/DC criterion, thus providing functions, their tests and a baseline coverage for further evaluations.

Acknowledgment

This work was supported by the Scientific Grant Agency of the Slovak Republic, grant No. VG 1/0725/19.

References

[1] Derek Fowler and Phil Bennett. 2000. IEC 61508 - A Suitable Bases for the Certification of Safety-Critical Transport-Infrastructure Systems? SAFECOMP '00. Springer-Verlag, 250–263.
[2] A. El-Serafy, G. El-Sayed, C. Salama and A. Wahba, "Enhanced Genetic Algorithm for MC/DC test data generation," 2015 INISTA, Madrid, 2015, pp. 1-8.
[3] *ISO 26262:2018 – Road vehicles -— Functional safety*, International Standardization Organization, Geneva, CH, 2018.
[4] S. Anand et al., "An orchestrated survey of methodologies for automated software test case generation," J. Syst. Softw., vol. 86, pp. 1978–2001, Aug. 2013.
[5] A. Pachauri and G. Srivastava, "Automated test data generation for branch testing using genetic algorithm: An improved approach using branch ordering, memory and elitism," J. Syst. Softw., vol. 86, pp. 1191–1208, 2013.
[6] P. Bokil, P. Darke, U. Shrotri, and R. Venkatesh, "Automatic test data generation for C programs," SSIRI 2009 - 3rd IEEE Int. Conf. Secur. Softw. Integr. Reliab. Improv., pp. 359–368, 2009.
[7] T. Su et al., "Automated coverage-driven test data generation using dynamic symbolic execution," Proc. - 8th Int. Conf. Softw. Secur. Reliab. SERE 2014, pp. 98–107, 2014
[8] P. Braione, G. Denaro, A. Mattavelli, and M. Pezzè, "Combining symbolic execution and search-based testing for programs with complex heap inputs," Proc. 26th ACM SIGSOFT Int. Symp. Softw. Test. Anal. - ISSTA 2017, pp. 90–101, 2017
[9] J. Kim, M. Kwon, and S. Yoo, "Generating test input with deep reinforcement learning," Proc. - Int. Conf. Softw. Eng., pp. 51–58, 2018
[10] Y. Jia, M. B. Cohen, M. Harman, and J. Petke, "Learning combinatorial interaction test generation strategies using hyperheuristic search," in Proc. - Int. Conf. on Software Engineering, vol. 1, pp. 540–550, 2015