On Measuring Combinatorial Coverage of Manually Created Test Cases for Industrial Software

Miraldi Fifo, Eduard Enoiu, Wasif Afzal
Email: mfo12002@student.mdh.se, eduard.enoiu@mdh.se, wasif.afzal@mdh.se
Software Testing Laboratory, Mälardalen University, Västerås, Sweden.

***Abstract*—Combinatorial coverage has been proposed as a way to measure the quality of test cases by using the input interaction characteristics. This paper describes the results of empirically measuring combinatorial coverage of manually created test cases by experienced industrial engineers working with embedded software development. We found that manual test cases achieve on average 78% 2-way combinatorial coverage, 57% 3-way coverage, 40% 4-way coverage, 20% 5-way combinatorial coverage and 13% for 6-way combinatorial coverage. These manual test cases can be augmented to achieve 100% combinatorial coverage for 2-way and 3-way interactions by adding eight and 66 missing test cases on average, respectively. For 4-way interactions, full combinatorial coverage can achieved by adding 658 missing test cases. For 5-way and 6-way interactions, full combinatorial coverage can be achieved by adding 5163 and 6170 missing test cases on average, respectively. The results of this paper suggest that manual test cases created by industrial engineers do not achieve a high combinatorial coverage and can be improved by adding more test cases to cover t-wise interactions at the expense of more test cases to execute.**

I. Introduction

In industrial practice, test cases are still created manually by using specific test design techniques and domain-specific skill and experience. Although criteria-based creation of test suites has been the focus of a great deal of research, manual testing is still widely used [1], [2] in the software development industry. In addition, in the industrial control software used in real time applications (e,g., trains, power plants), testing is not only rigorous but is also performed according to safety standards. It is of utmost importance to determine the right level of effectiveness for such test cases created manually by engineers. Combinatorial coverage [3], [4] has been proposed as a measure of test case quality based on t-way interaction measurement that can provide a quantitative measure of the input space combinations that have been tested.

This paper describes an empirical exploration of combinatorial coverage measurement of manual test cases created by industrial engineers working in Bombardier Transportation Sweden AB. The software programs of the tested systems are executed on Programmable Logic Controllers (PLCs). The manual test cases are created by experienced test engineers. In this paper we use the Combinatorial Coverage Measurement (CCM) [5][1] tool developed by NIST to analyze the thoroughness of test cases based only on the interaction coverage characteristics of a test case.

[1]https://github.com/usnistgov/combinatorial-testing-tools.git

The objective of this paper is to investigate the following research questions:

- RQ1: What is the combinatorial coverage achieved by test cases manually created by experienced engineers in industry?
- RQ2: What is the improvement potential in these test cases with respect to increased t-wise combinatorial coverage?

To answer these questions, we measure the combinatorial coverage of 33 test suites manually created by experienced industrial engineers. Each test suite contains a set of test cases. The measurements are performed with CCM tool and the results are evaluated in terms of the level of combinatorial achieved and the number of missing test cases to achieve full combinatorial coverage. The work introduced in this paper begins in Section II, where we describe the experimental setup used. Section III introduces the results on combinatorial coverage measurement of the manual test cases. We conclude with a summary of the threats to validity in Section IV and a brief discussion on the conclusions and implications of these results in Section V.

A. *Related Work and Background*

There exists several literature reviews on the topic of combinatorial testing and its applications [6]–[10]. Few previous studies [11]–[13] have focused on directly comparing manual test design techniques with automatic test generation in terms of test coverage, code coverage, and mutation analysis. However, these approaches are measuring the quality of the created test cases in terms of code coverage and fault detection. As a way to analyze the thoroughness of test cases based only on the test case characteristics, Kuhn et al. [14] introduced combinatorial coverage as a technique to measure the test quality. Combinatorial coverage measurements are based on the values of the input state space, thus this technique does not involve execution of the software.

In another study [4], researchers have described different measures of combinatorial coverage, which can help in the quality evaluation of a test set. Another study [3] presents useful combinatorial coverage methods used for estimating the coverage and quality of a test set in terms of fault detection for t-way interactions. In another study [15], combinatorial coverage is applied together with a new criterion to generate test suites. Instead of focusing on the interaction of different variables, in this case, the focus is on the interaction of

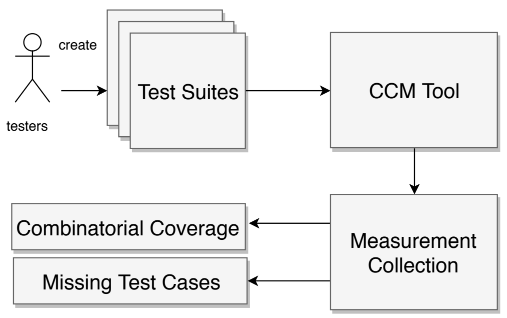

Fig. 1: Overview of the experimental method. For each created test suites we collect results on the combinatorial coverage achieved using the CCM tool.

different scenarios. This technique measures the quality of test cases based on a scenario-based coverage criterion. Recently, Vilkomir et al. [16] performed measurements for combinatorial coverage, three cryptographic algorithms are the object of this analysis and the CCM tool is used to measure the combinatorial coverage of random test cases. Since there is a lack of empirical studies measuring combinatorial coverage of manual test cases designed by experienced engineers, we are motivated to investigate this subject in more detail.

II. Experiment Setup

In this experiment, we perform measurements on 33 manual test cases, designed by experienced engineers, analyzing the collected results in terms of combinatorial coverage. We used the method shown in Figure 1. The test cases are provided by Bombardier Transportation Sweden AB, a large-scale company focusing on train and railway development. The system under test is a Train Control Management System (TCMS), an embedded system controlling the safety-critical functionality of a train implemented using Programmable Logic Controllers (PLCs) to provide control and supervision [17]. The PLC software running on this system is developed using a domain-specific programming language. PLC programs in TCMS are created using the IEC 61131-3 programming languages: *Structure Text* (ST), *Instruction List* (IL), *Ladder Diagram* (LD), *Function Block Diagram* (FBD). We note here that two of these languages, FBD and LD, are graphical programming languages. When a PLC program written in these languages is compiled to machine code, the program is evaluated using the read-execute-write semantics and executed cyclically using a predefined cycle time. PLC programs typically contain time-depended procedures that require input parameters to be triggered and changed at certain times in order to trigger specific behaviors. Each test case used in this study was created for testing a PLC program represented as a Program Organization Unit (POU) [17] containing functions (i.e., procedures), function blocks (i.e., stateful functions) and a top-level program that has access to the IO ports (e.g., sensors and actuators).

After performing measurements for 2 to 6 way interactions, we collect the generated results and use them to generate the statistics. The tool used in this paper to measure the combinatorial coverage of manual test cases is named Combinatorial Coverage Measurement (CCM) tool and it is developed at the National Institute Standards and Technology (NIST). When it comes to creating manual test cases, there are at least two ways [18] of creating test cases: criteria-based test design and human-based test design. The criteria-based design consists in designing test cases in order to satisfy some engineering requirements such as coverage criteria. On the other hand, the human-based test design is used for creating test cases mostly based on the testers knowledge about the software under test and the specific domain. Test cases used in this study are produced using both criteria and human-based test design according to safety-critical standards. In this paper we will focus on manual test cases created by engineers working in industry that have years of experience in both coverage-based and human experience-based test design.

TABLE I: Results for each t-way. We report several statistics relevant to the obtained results: minimum, median, mean, maximum and standard deviation values.

t-way	Median	Mean	Min	Max	SD
2-way	83%	78%	32%	100%	0.16
3-way	55%	57%	1%	100%	0.22
4-way	32%	40%	1%	100%	0.25
5-way	18%	20%	1%	50%	0.13
6-way	10%	13%	1%	40%	0.10

III. Results

In this section we present the results of the performed measurements. We collected the measurements data to answer our research questions by collecting manual test suites; measuring their effectiveness in terms of combinatorial coverage. The collected results are displayed as descriptive statistics (i.e. median, mean, minimum, maximum and standard deviation) and the overall results for the combinatorial coverage of each t-way interaction are displayed in Table I.

As can be noticed from the results in Table I, the majority of combinatorial coverage scores decrease while the t-way interaction increases. On average, the manual test suites achieve a combinatorial coverage of 78.6% for 2-way interactions, 57% in 3-way interactions, 40.2% in 4-way interactions, 20.2% in 5-way interactions and 13% in 6-way interactions on average for all test suites. In Figure 2, we depict the combinatorial coverage scores for each t-way interactions in the form of a boxplot; boxes span from 1st to 3rd quartile, black middle lines mark the median and the whiskers extend up to 1.5x the inter-quartile range and the circle symbols represent outliers.

The scatter plot in Figure 3 represents the distribution of the number of manual test cases and number of variables for each test suite. Each circle in the scatter plot represents a test suite, with the vertical axis showing the number of manual test cases for the specific test suite and the horizontal axis showing the number of variables for each particular test suite. We can observe that only eight out of all the 33 test suites (less than 25%) have more than 20 manual test cases, and only 11 out of the 33 test suites (33%) have more than 10 variables.

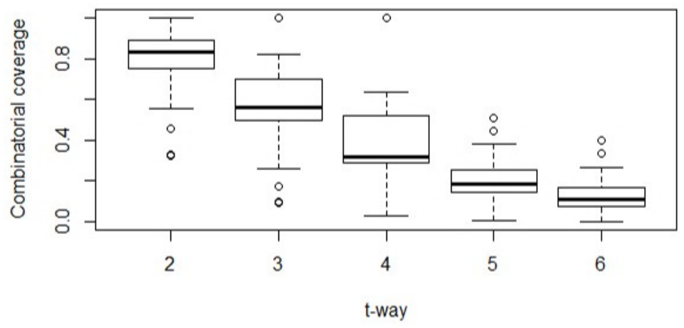

Fig. 2: Combinatorial coverage scores achieved for all t-way interactions.

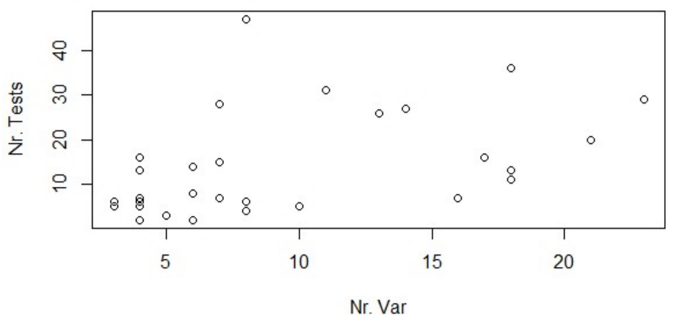

Fig. 3: Number of variables and the number of test cases in each test suite.

RQ1 asked what is the combinatorial coverage achieved by manually created test cases. Overall we confirm that manual test cases created by industrial engineers do not achieve very high combinatorial coverage.

Answer RQ1: Manual test cases cover less than 79% of input parameter interactions on average for t-way (where t is between 2 and 6). Manual test cases created by experienced engineers achieve 78.6% 2-way combinatorial coverage, 57% for 3-way coverage, 40.2% for 4-way coverage, 20.2% for 5-way coverage and 13% for 6-way coverage on average.

To answer RQ2, we collected data on the number of test cases needed for achieving 100% combinatorial coverage. This can be obtained by adding the missing test cases to the existing test suite, in order to cover the missing combinations and achieve full combinatorial coverage. The CCM tool provides the functionality to generate all missing test cases to achieve full combinatorial coverage. In Table II we show the results of the number of additional test cases on average needed for each t-way interactions to achieve full combinatorial coverage.

For example, for 2-way there are approximately eight missing test cases on average to be added to the existing test cases in order to achieve full combinatorial coverage. For 3-way there are approximately 66 missing test cases on average to be added to the existing test suites in order to achieve full combinatorial coverage. We note here that the maximum number of test cases in a test suite manually created is 4. On the other hand for 4-way interactions 658 missing test cases should be added to the existing ones in order to achieve full combinatorial coverage. For 5-way and 6-way interactions, we would need to add 5163 and 6170 missing test cases, respectively, to achieve full combinatorial coverage.

The CCM tool can generate at maximum 10000 missing test cases. For 2-way, 3-way and 4-way interactions there are no test suites where this limit is reached. For 5-way and 6-way interactions, in most of the test suites, this limit is reached. Therefore, for these cases adding these test cases becomes prohibitive from a test execution point of view. In Figure 4 we show the number of missing test cases for each t-way in form of a box-plot.

Answer RQ2: Manual test cases can be improved in terms of combinatorial coverage by generating additional test cases for 2-way and 3-way: 7.6 and 66.2 more test cases on average, respectively. For higher strength interactions, the improvement can be prohibitive due to the large number of missing test cases to achieve full combinatorial coverage (over 10000 missing test cases in most cases for 6-way).

IV. Validity Threats

The results are based on an experiment in one company in Sweden using a limited number of test suites. Even if this number can be considered small, we argue that having access to real qualitative data and the opportunity to collect information from engineers working in the embedded system domain can be representative. More case studies are needed to generalize these results to other systems and domains.

The assumption that the input state space for each test suite is determined by the values that each variable take in the manual test cases of the test suite might not reflect the real context. This is because some variables might take also other values that are not reflected in the existing test cases of the specific test suite. In order to avoid this validity threat, ranges of values should be used for each variable, including those values that are not represented in the existing test cases of the test suite.

Manual test cases in real life scenarios have constraints which define some combinations of variables as forbidden. These combinations are considered as invalid combinations and are excluded during the combinatorial testing analysis. In this paper we used test data without constraints. This aspect can be a validity threat to the generalization of the results to scenarios with constraints.

We used the CCM tool developed from NIST for combinatorial coverage analysis. This tool is based on well know algo-

TABLE II: The average number of needed test cases to achieve full combinatorial coverage on each t-way interaction for each existing test suite.

t-way	2-way	3-way	4-way	5-way	6-way
Test Cases	7.6	66.2	658	5163	6170

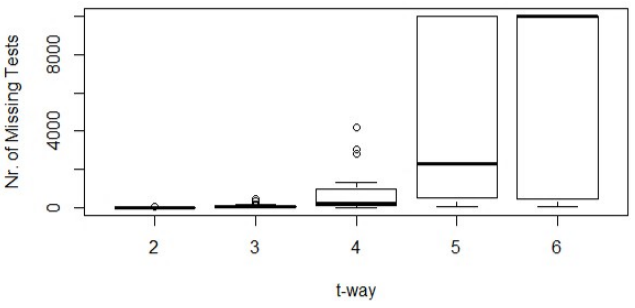

Fig. 4: Graphical representation of the number of missing test cases for each t-way interaction.

rithms and to the best of our knowledge is the only available tool that gives the possibility to generate the missing test cases to achieve the desired level of combinatorial coverage. We also assumed that its results are similar to the output produced by other combinatorial tools.

V. Conclusions and Future Work

In this paper we used the CCM tool to perform measurements of combinatorial coverage for 33 test suites, containing manual test cases designed by experienced engineers of Bombardier Transportation Sweden AB. The results showed that the level of the combinatorial coverage decreased as the strength of t-way interactions increased. More specifically, the achieved combinatorial coverage of manual test cases is 78.6% for 2-way interactions, 57% for 3-way interactions, 40.2% for 4-way interactions, 20.2% for 5-way interactions and 13% for 6-way interactions. The results imply that manual test cases written by industrial engineers do not achieve high combinatorial coverage scores regardless of the t-way combinations.

Additionally, the CCM tool allowed us to generate missing test cases to reach full combinatorial coverage for different t-way interactions. The results showed that manual test cases designed from experienced engineers can be improved by adding missing test cases to achieve full combinatorial coverage. For higher strength interactions, the improvement is infeasible to be useful in practice due to the large number of missing test cases (over 10000 missing test cases in most of the cases for 5-way and 6-way) needed to achieve full combinatorial coverage. Nevertheless, knowing the proportion of missing combinations could provide useful information for test engineers. The results of this study suggest that there might be a need to augment manually created test cases.

These results on measuring the combinatorial coverage of manual test cases need to be further studied; we need to consider fault coverage estimation as well as to consider the cost of using this measurements in practice.

References

[1] C. Andersson and P. Runeson, "Verification and validation in industry-a qualitative survey on the state of practice," in *International Symposium on Empirical Software Engineering*. IEEE, 2002, pp. 37–47.

[2] A. Beer and R. Ramler, "The role of experience in software testing practice," in *Euromicro Conference Software Engineering and Advanced Applications*. IEEE, 2008, pp. 258–265.

[3] D. R. Kuhn, R. N. Kacker, and Y. Lei, "Measuring and specifying combinatorial coverage of test input configurations," *Innovations in systems and software engineering*, vol. 12, no. 4, pp. 249–261, 2016.

[4] D. R. Kuhn, I. D. Mendoza, R. N. Kacker, and Y. Lei, "Combinatorial coverage measurement concepts and applications," in *International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2013, pp. 352–361.

[5] I. D. Mendoza, D. R. Kuhn, R. N. Kacker, and Y. Lei, "Ccm: A tool for measuring combinatorial coverage of system state space," in *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2013, p. 291.

[6] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Comput. Surv.*, vol. 43, no. 2, pp. 11:1–11:29, 2011.

[7] V. V. Kuliamin and A. A. Petukhov, "A survey of methods for constructing covering arrays," *Programming and Computer Software*, vol. 37, no. 3, p. 121, 2011.

[8] W. Afzal, R. Torkar, and R. Feldt, "A systematic review of search-based testing for non-functional system properties," *Inf. Softw. Technol.*, vol. 51, no. 6, pp. 957–976, 2009.

[9] R. E. Lopez-Herrejon, S. Fischer, R. Ramler, and A. Egyed, "A first systematic mapping study on combinatorial interaction testing for software product lines," in *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2015.

[10] B. S. Ahmed, K. Z. Zamli, W. Afzal, and M. Bures, "Constrained interaction testing: A systematic literature study," *IEEE Access*, vol. 5, pp. 25 706–25 730, 2017.

[11] E. Enoiu, D. Sundmark, A. Causevic, and P. Pettersson, "A comparative study of manual and automated testing for industrial control software," in *International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2017, pp. 412–417.

[12] P. Charbachi, L. Eklund, and E. Enoiu, "Can pairwise testing perform comparably to manually handcrafted testing carried out by industrial engineers?" in *International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 2017, pp. 92–99.

[13] G. Fraser, M. Staats, P. McMinn, A. Arcuri, and F. Padberg, "Does automated white-box test generation really help software testers?" in *Proceedings of the 2013 International Symposium on Software Testing and Analysis*. ACM, 2013, pp. 291–301.

[14] D. R. Kuhn, R. N. Kacker, and Y. Lei, "Combinatorial coverage as an aspect of test quality," *CrossTalk*, vol. 28, no. 2, pp. 19–23, 2015.

[15] V. P. La Manna, I. Segall, and J. Greenyer, "Synthesizing tests for combinatorial coverage of modal scenario specifications," in *18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 2015, pp. 126–135.

[16] S. Vilkomir, A. Alluri, D. R. Kuhn, and R. N. Kacker, "Combinatorial and mc/dc coverage levels of random testing," in *International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 2017, pp. 61–68.

[17] K.-H. John and M. Tiegelkamp, *IEC 61131-3: programming industrial automation systems: concepts and programming languages, requirements for programming systems, decision-making aids*. Springer Science & Business Media, 2010.

[18] P. Ammann and J. Offutt, *Introduction to software testing*. Cambridge University Press, 2016.