Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2020, 7(12): 78-83

Research Article ISSN: 2394-658X

Efficient Test Case Generation using Combinatorial Test Design: Towards Enhanced Testing Effectiveness and Resource Utilization

Kodanda Rami Reddy Manukonda

Email id - reddy.mkr@gmail.com

ABSTRACT

Combinatorial testing is a promising approach to software testing that aims to improve testing effectiveness and optimize resource utilization. It involves systematically exploring interactions among input parameters, generating a reduced set of test cases while maintaining adequate coverage. Empirical research shows that most software defects result from a few input parameter interactions, emphasizing the importance of adopting combinatorial testing methodologies. Automated combinatorial testing tools offer consistency, efficiency, and resource optimization in test case generation. However, the paper acknowledges limitations like the need for accurate parameter selection. Practical examples demonstrate its effectiveness in reducing testing times and costs. The paper also provides insights into combinatorial test design algorithms and tools, including the Advanced Combinatorial Testing System (ACTS).

Keywords: Combinatorial testing, software testing, test case generation, input parameter interactions, resource utilization, testing effectiveness, automated testing tools

INTRODUCTION

A relatively recent development in software testing is combinatorial testing. It is predicated on straightforward ideas like pairs testing, t-way interaction testing, interaction rule, etc. Combinatorial testing, which lowers the number of tests (i.e., test cases) that must be run, can significantly save testing expenses and times [1]. Even dependable software can occasionally malfunction when combination of inputs entered is old. Testing every combination beforehand where a lot of possible inputsbecomes a tedious operation. In many cases, this type of thorough testing is no longer possible because modern software may need hundreds of inputs. The bulk of defects are caused by the interplay of a few inputs, according to empirical study. Combinatorial testing enables us to choose two or three configurations or inputs at once and test the combinations in a way that will reveal the majority of the errors with the fewest number of tests. Rule of interaction. The majority of software malfunctions are caused by errors in one parameter or by the interaction of two. As can be seen here, an increasing number of parameters interact to cause progressively fewer problems.

A. Need of Combinatorial testing

Combinatorial testing tools are user-friendly test case generators that enable the input and constraints to be provided to the input parameter model, after which the model is used to build test configurations.

[1]. **Consistency:** Automated combinatorial testing solutions enable uniform creation, execution, and evaluation of test cases. This consistency reduces the chance of human error while ensuring that testing is done methodically and consistently.

[2]. **Efficiency:** Combinatorial testing tools automate the process of creating test cases by closely examining the possible combinations of input parameters and their values. This approach is more effective than manual testing and can quickly cover a large number of test scenarios.

[3]. **Resource Optimization:** Combinatorial testing tools aid in the optimization of testing resources by focusing on the most significant combinations of input parameters.

[4]. **Coverage:** These techniques help identify potential faults resulting from parameter interactions by offering comprehensive coverage of all conceivable input parameter combinations.

[5]. **Risk Reduction**: Combinatorial testing approaches systematically examine combinations of input parameters, thereby lowering the likelihood of software errors caused by unexpected interactions or dependencies between parameters.

[6]. **Time Savings:** Automated combinatorial testing methods reduce the amount of time testers must spend building and managing large sets of test cases by automatically producing test cases.

B. Disadvantages of Combinatorial testing

[1]. In combinatorial testing, test setups are meaningless if values for input variables are not selected correctly.

[2]. In a similar vein, the preceding point holds true for choosing and implementing the limitations.

[3]. Inadequate comprehension of the input parameters may lead to an improper combinatorial object array.

[4]. Performing combinatorial testing by hand is an expensive and time-consuming operation.

[5]. Even with automated tools, creating test settings is a time-consuming procedure.

LITERATURE REVIEW

Chen et al. [4] Gives a thorough explanation of metamorphic testing, a software testing method that makes use of relationships that are constant between input and output properties during several executions [4]. It emphasizes how crucial metamorphic testing is for finding errors that more conventional approaches would have overlooked. Issues including finding appropriate metamorphic relations, writing efficient test cases, and automating the testing procedure are covered in the review. Additionally, it draws attention to ways to advance metamorphic testing, like incorporating machine learning methods and investigating non-traditional uses for it. The review provides insightful information for next studies.

Nižetić et al. [5] Examines the use of smart technologies in waste management, sustainable resource use, and energy efficiency. It highlights how crucial these technologies are to solving environmental problems and enhancing resource efficiency. The study covers the basic ideas of smart technologies and how they can be used to reduce energy use, encourage the sustainable use of resources, and enhance waste management. In order to improve performance and scalability, it also highlights new developments in smart technologies, such as the fusion of IoT, data analytics, and AI. The review offers insightful information about how smart technology might improve waste management, resource efficiency, and energy efficiency.

Papadakis et al.'s [6] Offers a thorough examination of mutation testing, a software testing method used to gauge the efficacy of test suites. The writers go over the fundamental ideas of mutation testing, talk about new developments in methodology, and look at case studies and empirical research showing how effective it is at raising the caliber of software. They also discuss difficulties and constraints like the requirement for efficient mutation detection techniques and processing costs. The report establishes the foundation for future research to improve software testing processes and offers insightful information on the development of mutation testing approaches.

Shi et al.'s [7] Talk about the difficulties in reducing N2 and emphasize the significance of creating effective catalysts. They examine current advancements in high-throughput screening methods, sophisticated characterization techniques, and computational modeling in rational catalyst design approaches. The review also looks at important elements including surface modification, active site engineering, and synergistic effects that affect the catalytic activity and selectivity of N2 reduction catalysts. The review offers insightful information about the state of catalyst design at the moment and presents viable methods for obtaining improved conversion efficiency in ambient environments.

FORMS OF COMBINATORIAL TESTING

Configurations and/or inputs can be considered as parameters in combinatorial testing. Input parameter testing and configuration testing are the names of the two techniques [8]. To gain a better understanding of how combinatorial testing operates, view the example provided below.

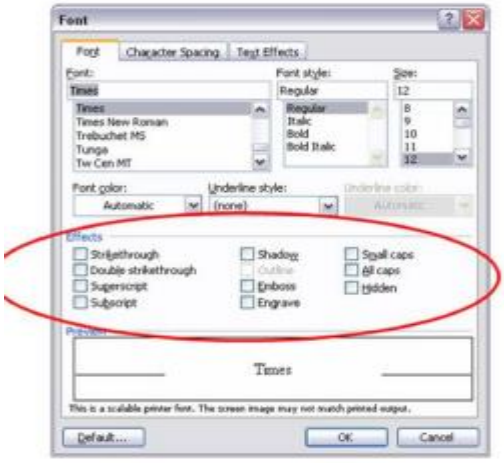

Figure 1: A dialog box with 10 options

A word processing software's font formatting dialog is displayed in Figure 1. There are ten effects that can be turned on or off, many of which are independent of one another. As a result, 210 (=1024) combinations exist. It is expensive to test each of these combinations. Reducing the number of tests to be conducted is the method used in combinatorial testing, as was previously discussed. Assuming that three-way interactions will disclose every error, let's determine how many tests to run.

Number of tests $=\frac{10}{3}$ = 120 tests.

There are 23 possible outcomes when each triplet (the three parameters chosen for a test) is turned on or off. 120 x 23 = 960 tests are involved.

The maximum number of tests is 320 since, if 10 inputs are chosen, three triplets can be used in a single test.

But in practice, many triplets can be packed into one test

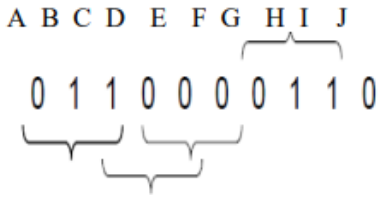

Figure 2: A single test case containing many triplets

Examine Figure 2. ABC, DEF, and GHI are the three triplets that appear to be in the test case. In actuality, though, this collection of inputs contains a lot more triplets [9]. As an illustration, BCD, EFG, and HIJ can all be thought of as additional triplets. ABF, ABE, and ABD are also. With an example test array of 13 test instances, let's examine this phenomenon in greater detail.

A	B	C	D	E	F	G	H	I	J
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	0	0	0	0	1
1	0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	1	0	0	0
0	1	1	0	0	1	0	0	1	0
0	0	1	0	1	0	1	1	1	0
1	1	0	1	0	0	1	0	1	0
0	0	0	1	1	1	0	0	1	1
0	0	1	1	0	0	1	0	0	1
0	1	0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	1	1	1
0	1	0	0	0	1	1	1	0	1

Figure 3: Arrangement of test cases

Let's determine the bare minimum of tests necessary to achieve the configuration of test cases shown in Figure 3

A	B	C	D	E	F	G	H	I	J
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	0	0	0	0	1
1	0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	1	0	0	0
0	1	1	0	0	1	0	0	1	0
0	0	1	0	1	0	1	1	1	0
1	1	0	1	0	0	1	0	1	0
0	0	0	1	1	1	0	0	1	1
0	0	1	1	0	0	1	0	0	1
0	1	0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	1	1	1
0	1	0	0	0	1	1	1	0	1

Figure 4: Arrangement of a single triplet

Examine Figure 4. Three effects are depicted in the first three columns (A, B, C) [10]. The above table contains all eight possible combinations with these three inputs.

A	B	C	D	E	F	G	H	I	J
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	0	0	0	0	1
1	0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	1	0	0	0
0	1	1	0	0	1	0	0	1	0
0	0	1	0	1	0	1	1	1	0
1	1	0	1	0	0	1	0	1	0
0	0	0	1	1	1	0	0	1	1
0	0	1	1	0	0	1	0	0	1
0	1	0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	1	1	1
0	1	0	0	0	1	1	1	0	1

Figure 5: Arrangement of two triplets

Let's now examine Figure 5. Three columns—D, E, and G—have been arbitrarily chosen in this instance (light gray circles). Included as well are the eight possible combinations with those three inputs.

A	B	C	D	E	F	G	H	I	J
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	0	0	0	0	1
1	0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	1	0	0	0
0	1	1	0	0	1	0	0	1	0
0	0	1	0	1	0	1	1	1	0
1	1	0	1	0	0	1	0	1	0
0	0	0	1	1	1	0	0	1	1
0	0	1	1	0	0	1	0	0	1
0	1	0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	1	1	1
0	1	0	0	0	1	1	1	0	1

Figure 6: Arrangement of three triplets

Examine Figure 6. Here, a test case is represented by each row and a parameter by each column. Upon further inspection, it is evident that there are numerous triplets present, not just three. These 13 rows contain every possible combination of the 10 parameters [11]. Thus, we can conclude that only 13 tests are needed as a minimum. Compare this to the 1024 comprehensive tests or the estimated 120 tests.

Covering array: The challenging aspect of combinatorial testing is creating the appropriate set of test scenarios. We can determine the bare minimum of tests necessary to cover all t-way combinations by appropriately organizing the test cases (3-way, in the preceding example). The term "covering array" refers to this configuration of test cases and parameters [12]. The covering array can be produced quickly by automated testing techniques that use combinatorial testing.

CONSIDERATIONS FOR COMBINATORIAL TESTING AND TEST TOOLS

When there are numerous parameters (inputs/configurations) and errors arise from the interplay of these elements, combinatorial testing is the most appropriate option [13]. It might be used if the system in question is crucial yet comprehensive testing isn't feasible because of specific conditions.

When there are very few parameters for which exhaustive testing is feasible, combinatorial testing is useless. When there are no 41 relationships between parameters, it is also meaningless. In a similar vein, where interactions between various inputs/configurations do not result in new errors, combinatorial testing should not be used.

A. Test Tools Incorporating Combinatorial Testing

An algorithm known as combinatorial test design (CTD) finds a simple test plan that encompasses all possible interaction levels to 100%. The National Institute of Standards and Technology (NIST) has released five new combinatorial testing tools.

[1]. Advanced Combinatorial Testing System (ACTS): This system creates test sets with variable-strength tests and support for constraints, ensuring t-way coverage of input parameter values.

[2]. Calculates several coverage measures of an existing test set using combinatorial coverage measurement.

[3]. The Access Control Policy Test (ACPT) generates tests for access control policies by combining model checking and combinatorial testing.

[4]. For.NET systems with a lot of configuration variables, the NET Configuration test file generator offers combinatorial coverage.

[5]. Combinatorial sequence test generator: this tool creates sequences covering arrays and is helpful for testing hardware, protocols, and GUIs in event-driven systems.

CONCLUSION

Combinatorial testing is a strategic approach to enhancing testing effectiveness and optimizing resource utilization in software development processes. It involves systematically exploring interactions among input parameters, addressing the increasing complexity of modern software systems [14]. By utilizing principles like pairwise and t-way interaction testing, combinatorial testing generates a reduced set of test cases while ensuring comprehensive coverage of potential faults. Empirical research highlights the importance of combinatorial testing in identifying and mitigating software defects arising from input parameter interactions [15]. However, it requires accurate selection of input parameter values and constraints. The effectiveness of combinatorial testing relies on the availability of automated tools for efficient test case generation and configuration management. The National Institute of Standards and Technology (NIST) offers resources to support combinatorial testing practices in software development projects.

REFERENCES

[1]. K. D. Davis et al., "Discovery and validation of biomarkers to aid the development of safe and effective pain therapeutics: challenges and opportunities," Nat. Rev. Neurol., vol. 16, no. 7, pp. 381-400, 2020.

[2]. T. Y. Chen et al., "Metamorphic testing: A review of challenges and opportunities," ACM Comput. Surv., vol. 51, no. 1, pp. 1-27, 2018.

[3]. S. Nižetić et al., "Smart technologies for promotion of energy efficiency, utilization of sustainable resources and waste management," J. Clean. Prod., vol. 231, pp. 565-591, 2019.

[4]. M. Papadakis et al., "Mutation testing advances: an analysis and survey," Adv. in Computers, vol. 112, pp. 275-378, 2019.

[5]. L. Shi et al., "Rational catalyst design for N2 reduction under ambient conditions: strategies toward enhanced conversion efficiency," ACS Catal., vol. 10, no. 12, pp. 6870-6899, 2020.

[6]. K. Chakrabarty and F. Su, Digital Microfluidic Biochips: Synthesis, Testing, and Reconfiguration Techniques. CRC Press, 2018.

[7]. F. Tao et al., "Digital twin-driven product design, manufacturing and service with big data," Int. J. Adv. Manuf. Technol., vol. 94, pp. 3563-3576, 2018.

[8]. K. Krebs and L. Milani, "Translating pharmacogenomics into clinical decisions: do not let the perfect be the enemy of the good," Hum. Genomics, vol. 13, pp. 1-13, 2019.
[9]. T. Rakha and A. Gorodetsky, "Review of Unmanned Aerial System (UAS) applications in the built environment: Towards automated building inspection procedures using drones," Autom. Constr., vol. 93, pp. 252-264, 2018.
[10]. Q. Yang et al., "Investigating how experienced UX designers effectively work with machine learning," in Proc. 2018 Designing Interactive Systems Conf., pp. 585-596, Jun. 2018.
[11]. Bargaz et al., "Soil microbial resources for improving fertilizers efficiency in an integrated plant nutrient management system," Front. Microbiol., vol. 9, p. 1606, 2018.
[12]. S. Sengupta et al., "A review of deep learning with special emphasis on architectures, applications and recent trends," Knowl.-Based Syst., vol. 194, p. 105596, 2020.
[13]. M. Zekić-Sušac, S. Mitrović, and A. Has, "Machine learning based system for managing energy efficiency of public sector as an approach towards smart cities," Int. J. Inf. Manag., vol. 58, p. 102074, 2021