iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** XII **Month of publication:** December 2018

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue XII, Dec 2018- Available at www.ijraset.com

Literature Survey on Combinatorial Interaction Testing for Large Complex Real Time Embedded Systems

P Venkata Sarla[1], Dr. Balakrishnan Ramadoss[2]
[1]*Computer Science Department, Bharathiar University, Coimbatore, India*
[2]*NIT Trichy*

***Abstract:** This paper presents literature survey on various aspects of combinatorial interaction testing. Various techniques of generating combination covering arrays using a wide range of algorithms are studied. The survey includes study of identification of benefits and limitations of this testing approach, applicability of this testing strategy for systems of various domains, analysis of need for handling constraints, methods for generation of test cases for combinatorial interaction testing and various tools for generation of test data. In spite of the huge benefits of CIT, reasons for not applying this testing strategy for large complex real time embedded systems are explored and presented.*

***Keywords:** Combinatorial Interaction Testing, Combination Strength, Constraints, Coverage Criteria, Real Time Embedded Systems, Software Requirement Specifications*

I. INTRODUCTION

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. Interaction failures as explained in [1] are one of the primary reasons why software testing is so difficult. If failures only depended on one variable value at a time, we could simply test each value once, or for continuous-valued variables, one value from each representative range or equivalence class. If our application had inputs with V values each, this would only require a total of V tests one value from each input per test. But in real world, the software systems are much complicated involving interactions within parameters.

Combinatorial Interaction Testing (CIT), which has proven very effective in fault detection, is a testing strategy that applies the theory of combinatorial design to test such software systems. Given a system under test with k parameters, t-way combinatorial testing requires all combinations of values of t (out of k) parameters be covered at least once. If test parameters are modelled properly, all faults caused by interactions involving no more than t parameters will be detected. Combinatorial testing can significantly reduce the cost of software testing while increasing its effectiveness.

However, from the study it is found that CIT is not applied for large complex real time embedded systems. The reasons for the same are explored by studying the specific requirements of complex avionics systems used in modern combat aircrafts and helicopters.

II. LITERATURE SURVEY

In [1] The Interaction- rule states that most failures are caused by one or two parameters interacting, with progressively fewer failures caused by 3, 4, or more parameter interactions as depicted in Fig 1. So testing combinations of these values only up to an appropriate level is likely to catch nearly all errors.

However, in safety critical and mission critical avionics software systems, there will be many functions which involve interaction between more numbers of variables. CIT of any functionality implemented should ensure combination coverage with highest strength for that functionality.

The level of interaction between the parameters has to be captured by analysis of requirements and accordingly the strength t needs to be arrived at. In [1] and [4], prioritisation of running the test cases for testing higher priority functionalities earlier during the test phase is covered. For CIT, most of the researchers have taken programs from Software-artefact Infrastructure Repository (SIR) as subjects as mentioned in [4]. In [3], [5], [6], [7], [8], [9] and [10] the authors highlighted various aspects of input constraints in CIT for highly-configurable software like operating systems, development environments, mobile phone, telephone billing system, cruise controller, Model Checker and Compiler

©IJRASET: All Rights are Reserved

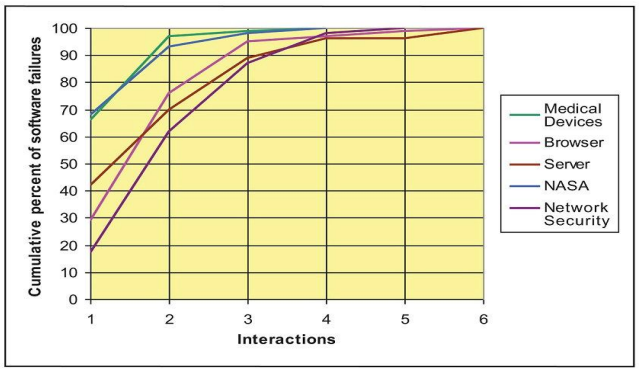


Fig. 1 Cumulative percentage of failures triggered by t-way interactions

In [2], the authors have used programs from Software-artefact Infrastructure Repository (SIR) as their subjects for examining the effectiveness of CIT on regression testing and test case prioritization. In [3] development of Test Case Generation (TCG) algorithm for CIT and idea for considering input constraints and building a unit testing harness from TCG is addressed. In [3] the authors have done systematic literature study on constrained interaction testing and discussed the applications of CIT in several domains, such as software product lines, fault detection and characterization, test selection, security, and graphical user interface testing. It is mentioned that Constrained Interaction Testing is an upcoming research direction in interaction testing. In most CIT applications, the problem domain is constrained. Some interactions are simply infeasible due to these constraints as explained in [6] and [11]. The nature and description of such constraints is highly domain specific. Any CIT approach that fails to take account of constraints will produce many test cases that are either unachievable in practice or which yield expensively misleading results. In [4], the authors have explained the specification based weight assignment method for prioritisation of combinatorial test cases. In [6], the authors illustrated that adding constraints in CIT of highly configurable systems, reduces the number of feasible system configurations but it is not guaranteed to reduce the size of the CIT sample to achieve coverage of desired strength. SPIN Model Checker software and GCC compiler software were taken as the case studies. In [7], it is mentioned that real systems are typically constrained and their constraints must be accounted for to avoid generation of inapplicable or misleading test cases. Reference [8] covers discussion on integrated approach for finding covering arrays and application of the same for constructing variable strength arrays. Not much research has happened in the usage of CIT for avionics systems. The authors in [12] discussed the test case generation for spacecraft mission software with simple input constraints and automated testing harness which will have a test result evaluator also. However, need for redefining the input space based on the constraints on outputs is not discussed. Variety of Combination strategies are discussed in [9] [13] [14]. But required combinations of input and intermediate parameters needed for effective CIT of avionics systems are not discussed in any references. In [9], Orthogonal array based testing strategy (OATS) is specified as the combinatorial testing method for testing pair-wise interactions. OATS technique is explained to be suitable for products with a large number of configuration possibilities by taking the example of mobile phone applications testing. In [16] The software of ACTS tool which itself is a useful for generating covering arrays for CIT of various types of systems, is taken as case study for testing the effectiveness of ACTS tool. This means, ACTS tool itself is used to test the effectiveness of the software implementation of ACTS tool. In [12], algorithm for automating of test case generation for Deep Space One mission's Remote Agent Experiment is explained. The survey carried out in [13] demonstrates a wide variety of combination strategies and reports on their usages. All the examples taken in this survey are on small software projects as compared to complex real time embedded systems. In [14], comparative evaluation of five combination strategies for generation of covering arrays for CIT, viz., All Combination Strategy (AC), Each Choice Strategy (EC), the Base Choice Strategy (BC), Orthogonal Arrays (OA) and the algorithm from the Automatic Efficient Test Generator (AETG). In [18], Base choice method is taken as reference and new strategies called Constrained Base Choice (CBC) and Extended Constrained Bases Choice (ECBC) are evolved. These new strategies are explained by applying them on two applications called NextDate and Ericsson which are of about 179 and 694 lines of code respectively. In [15], study of a number of tools and the covering array generation algorithms used in the tools on the basis of techniques and coverage strengths is carried out The results of the study can be of interest to researchers or software companies looking for combinatorial test algorithms or tools suitable for their testing needs. In [17] an approach to automate unit and integrating testing of radio's control software is described. In [19], the authors have illustrated an automated approach for finding and fixing conformance faults between given software system and its combinatorial model. In [20] automatic test data generation for testing of C programs at white box level for obtaining multiple

©IJRASET: All Rights are Reserved

coverage criteria including MCDC is covered. In [21], the authors have discussed about the extent of statement/branch and MC/DC coverage and the Fault Detection Rate (FDR) that can be achieved by executing CIT cases with strength varying from 2 to 5 on two subjects taken from SIR. They have taken original implementation and number of mutants of the subjects to study the effectiveness of CIT. Though they have generated covering arrays with strength of 5, as executing all the test cases with higher combination strength is a huge effort, the number of test cases that were executed is limited to that with strength 4. But a decrease in the statement/branch/MCDC coverage and FDR was noted by running only a subset of test cases with higher strength when compared to execution of complete set of test cases with lower strength. In [22] automatic generation of test configurations that cover all pair-wise interactions using feature models for testing Software Product Line (SPL) is explained. In [23] the authors have proposed a framework for automated pair-wise testing of SPL, with an objective to generate the minimal set of test configurations that are valid and cover all pair-wise feature interactions. Use of combinatorial approaches to develop test cases that systematically test important components of database backed web applications is demonstrated in [24]. Application of combinatorial testing to string text searches in the US National Vulnerability Database, a system that is accessed more that 70 million times a year is taken as case study. Conventional testing methods often rely on a “more is better” heuristic without an ability to measure completeness. Using covering arrays, or measuring combinatorial coverage of random tests, provides a sound test engineering method with defensible, quantitative measures of test completeness.

III. INFERENCES

From the study of available literature on CIT, it is observed that Combination Interaction Testing is not applied for large complex real time embedded systems used in avionics systems and automobile industry. From the experience of requirements management, design and testing methodologies used for complex avionics systems, the reasons for not applying the CIT approach for these systems is because of the following reasons:

1) The size of software of these systems is too large of the order of thousands of requirement specifications. The source code is about one to ten millions of Lines of Code (LOC). The number of test cases shall be of the order of tens of thousands.

2) For each functionality, these systems receive multiple inputs from various other systems, process these inputs and generate multiple outputs which are updated to number of interfacing equipment for consumption. For e.g., for mission critical requirements for a particular mode of combat implemented in a mission computer system of a combat aircraft, there will be about thirty to forty input parameters. These input parameters are received on different interfaces like MIL-1553B, RS232, RS422, RS485, discrete, analog and Ethernet. After processing these inputs the system generates multiple outputs which are of different formats and data types as required to be updated to number of other systems interfacing on different buses.

3) These systems have many mission critical and safety critical requirements and the software is developed using standards like MIL-STD-2167A, IEEE 12207, DO-178B/ 178C. For mission critical and safety critical avionics software systems, it is essential to achieve 100% statement/branch and MC/DC coverage so that the FDR also is high. These coverage criteria are addressed in the above standards but none of these standards have specified about the combination coverage requirements.

4) The existing tools do not cater for combination coverage of multiple mixed strengths with complex combinations as required for these types of systems.

5) The existing tools do not support handling of floating point inputs fed as constraints. But for these systems most of the input and output data is of floating point type with very high resolution.

IV. SCOPE FOR RESEARCH

As Combinatorial Interaction Testing has significant advantages, it can be applied to large complex real time embedded systems used in avionics systems of modern combat aircrafts and helicopters. However, due to the reasons mentioned above, it may be difficult to prepare and execute combinatorial interaction test cases for systems of these types. There is scope for research on the following aspects of applying CIT for systems of this type.

1) Identify the implications of applying CIT for these systems.

2) Identify the limitations of the tools for generating covering arrays with the typical constraints and mixed combination strengths specific to these systems.

3) Evolve the methods to mitigate the implications of applying CIT for these systems.

4) Identify and apply new combination strategies for effective CIT of these systems.

5) Develop tools for applying new combination strategies and generating covering arrays with complex combinations of strengths.

6) Develop covering array generation tools for handling floating point numbers in constraints.

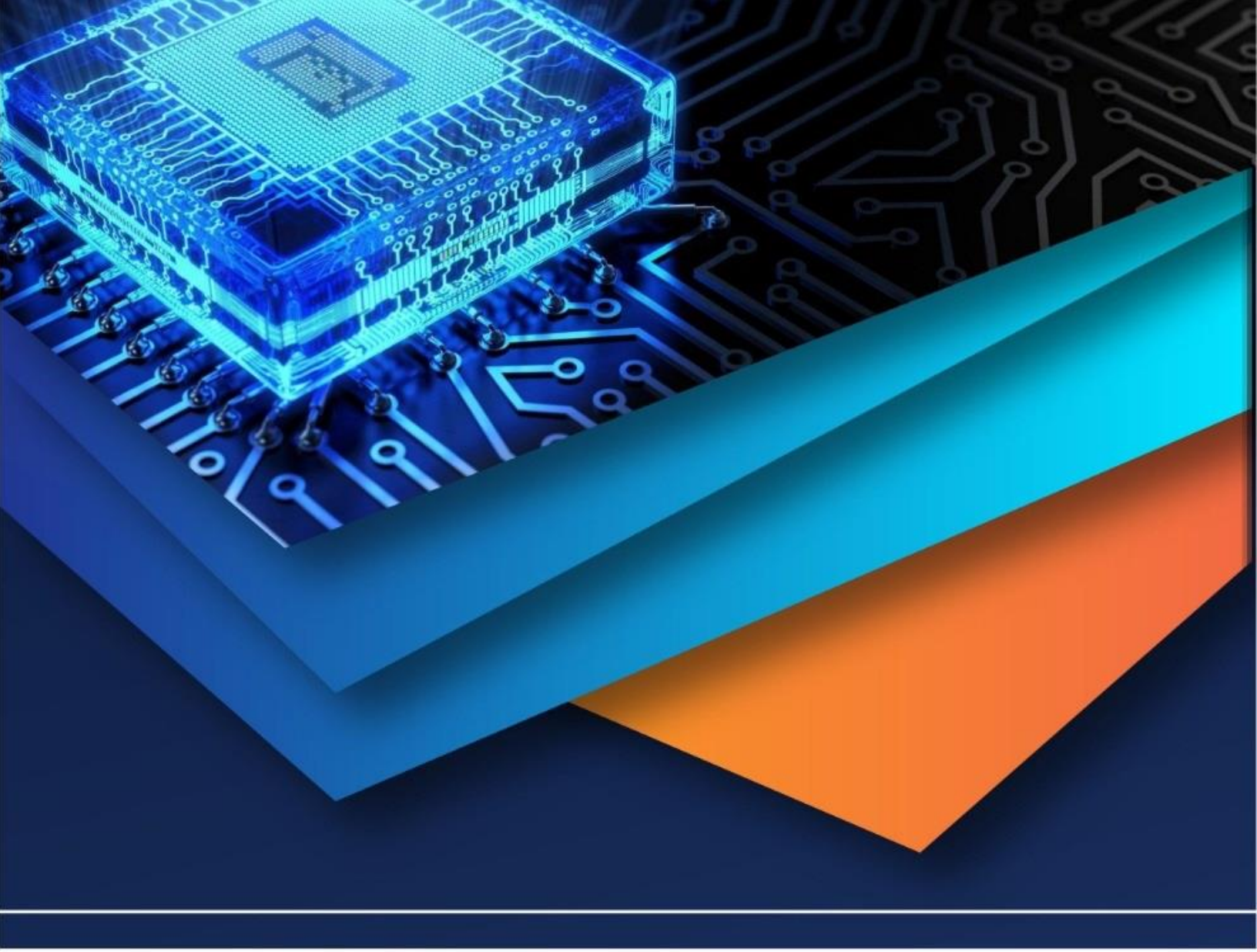
7) Automate the process of CIT for these systems.

©IJRASET: All Rights are Reserved

REFERENCES

[1] D. R. Kuhn, R. N. Kacker and Y. Lei, Introduction to Combinatorial Testing, Chapman & Hall/CRC Press, 2013.
[2] X. Qu, M. B. Cohen and K. M. Woolf, "Combinatorial Interaction Regression Testing: A Study of Test Case Generation and Prioritization," IEEE ICSM, 2007.
[3] B. S. Ahmed, Z. K. Zamli, W. Afzal and M. Bures, "Constrained Interaction Testing: A Systematic Literature Study," IEEE Access, vol. 5, p. 25206 to 25730, 2017.
[4] M. Aggarwal and S. Sabharwal, "Prioritisation Techniques in Combinatorial Testing: A Survey," IEEE, 2016.
[5] C. Nie and H. Leung, "A survey of Combinatorial Testing," ACM computing Surveys, vol. 43, p. 11.1 to 11.29, 2011.
[6] M. B. Cohen and M. B. Dwyer, "Constructing interaction test suites for highly configurable Systems in the presence of constraints: A greedy Approach," IEEE Transactions on software engineering, vol. 34, 2008.
[7] J. Petke, M. Cohen, M. Harman and S. Yoo, "Efficiency and early fault detection with lower and higher strength combinatorial interaction testing," ACM, 2013.
[8] M. B. Cohen and C. J. Colbourn, "Constructing Test Suites for Interaction Testing," Proceedings of IEEE 25th International Conference on Software Engineering (ICSE), 2003.
[9] R. Krishnan, S. M. Krishna and P. S. Nandhan, "Combinatorial Testing: Learnings from our Experience," ACM SIGSOFT Software Engineering Notes, vol. 32, p. 1 to 8, 2007.
[10] A. Calvagna and A. Gargantini, "A Formal Logic Approach to Constrained Combinatorial Testing," Journal of Automated Reasoning , Vols. DOI: 10.1007/s10817-010-9171-4, p. 1 to 26, 2010 .
[11] J. Petke, "Constraints: the Future of Combinatorial Interaction Testing," IEEE/ACM 8th International Workshop on Search-Based Software Testing, p. 17 to 18, 2015.
[12] Y. W. Tung and W. S. Aldiwan, "Automating Test Case Generation for the New Generation Mission Software System," 2010.
[13] M. Grindal, J. Offutt and S. F. Andler, "Combination Testing Strategies: A Survey," GMU Technical Report ISE-TR-04-05, 2004.
[14] M. Grindal, B. Lindstrom, J. Offutt and S. F. Andler, "An Evaluation of combination strategies for test case selection," GMU, 2006.
[15] S. K. Khalsa and Y. Labiche, "An Orchestrated Survey of available algorithms and tools for Combinatorial Testing," Research Gate, 2014.
[16] M. N. Borazjany, L. Yu, Y. Lei, R. Kacker and R. Kuhn, "Combinatorial Testing of ACTS: A Case Study," IEEE Fifth International Conference on Software Testing, Verification and Validation, p. 591 to 600, 2012.
[17] R. Bartholomew, "An Industry Proof-of-concept Demonstration of Automated Combinatorial Test," IEEE AST, p. 118 to 124, 2013.
[18] S. K. Khalsa and Y. Labiche, "An Extension of Category Partition Testing for Highly Constrained Systems," IEEE 17th International Symposium on High Assurance Systems Engineering, p. 47 to 54, 2016.
[19] A. Gargantini, J. Petke and M. Radavelli, "Combinatorial Interaction Testing for Automated Constraint Repair," 10th IEEE International Conference on Software Testing, Verification and Validation Workshops, p. 239 to 248, 2017.
[20] Prasad Bokil, Priyanka Darke, Ulka Shrotri, R Venkatesh, "Automatic Test Data Generation for C Programs" Third IEEE International Conference on Secure Software Integration and Reliability Improvement, 2009.
[21] Dong Li, Linghuan Hu, Ruizhi Gao, W Eric Wong, D. Richard Kuhn, Raghu N. Kacker, "Improving MC/DC and Fault Detection Strength Using Combinatorial Testing", IEEE Internatoinal Conference on Software Quality, Reliability and Security 2017.
[22] Aymeric Hervieu, Benoit Baudry, Arnaud Gotlieb, "PACOGEN: Automatic Generation of Pair-wise Test Configurations from Feature Models", Proceeding SOF International Symposium on Software Reliability Engineering (ISSRE'11), 2011.
[23] Dusica Marijan, Arnaud Gotlieb, Sagar Sen, Aymeric Hervieu "Practical Pairwise Testing for Software Product Lines" SPLC August 2013.
[24] M S Raunak, D Richard Kuhn, Raghu Kacker, "Combinatorial Testing of Full Text Search in Web Applications", IEEE International Conference on Software Quality, Reliability and Security 2017.

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)