# Combined Classification Tree Method for Test Suite Reduction

B. Ramadoss
Professor and Head
Department of Computer Applications
National Institute of Technology
Tiruchirappalli – 620 015

P. Prema
Research Scholar
Department of Computer Applications
National Institute of Technology
Tiruchirappalli – 620 015

S. R. Balasundaram
Associate Professor
Department of Computer Applications
National Institute of Technology
Tiruchirappalli – 620 015

## ABSTRACT
In software testing, test case generation is an important activity to ensure software quality. Various test case generation techniques are being provided for high reliable software systems. It becomes critical to execute all the test cases in certain time period. In order to reduce the number of test cases, test suite reduction techniques with respect to some coverage criteria are used. This paper provides the significance of classification tree method and combined classification tree method over equivalence class partition method for test case generation. This paper presents a comparison of these test case generation methods to show that which method has good potential to use for test suite reduction.

## General Terms
Measurement, Design, Verification.

## Keywords
Classification tree method, Equivalence class partition method, Combined classification tree method, Coverage criteria, Test case generation, Test Suite Reduction

## 1. INTRODUCTION
In software development life cycle, testing is necessary to produce highly reliable applications related to space, avionics, etc. Software developers have responded to this need in various ways through improving the process, increasing the attention on early development activities, and using formal methods for describing requirements, specifications, and designs. Even than it is not sufficient to define quality of the delivered software and to evaluate such testing, precise criteria is required [1].

Designing test cases is the most challenging part of testing and is used to detect defects in the developed software [2]. Test cases are usually generated based on application as well as the specification. In case of specification based testing, test cases are generated in the earlier stages of software development. Test cases are more effective for making the test plan [18]. This paper concentrates on classification tree method (CTM) and partition based test case generation methods as they are widely used for large scale and complex software systems.

Test suite reduction is an important activity for test maintenance that attempts to remove redundancy reduce execution time and thus decrease the testing cost. It reduces the number of test cases with respect to some coverage criteria defined as a set of rules which helps to determine whether a test suite has been adequately tested the software or not [7, 12].

This paper also discusses percentage of test suite reduction for various test case generation methods such as Equivalence Class Partition Method (ECPM), In-house and External Classification Tree Methods, and Combined Classification Tree Method (CCTM).CCTM is proposed for reducing redundancy test case generation.

## 2. Background
### 2.1 Test Case Generation Methods
#### *2.1.1 Equivalence Class Partition (ECP) method*
ECP method [10] is one of the test case generation methods and is used to find the minimum set of test data for valid and invalid inputs. Tester can subdivide the input domain into a relatively small number of partitions or disjoint sub domains which will be used for selecting the test data.

The guidelines for generating test conditions or equivalence classes based on ECPM [10], [11] are shown in Table.1. After identifying test conditions, one test case is generated for all valid class and separate test cases are generated for each invalid input.

#### *2.1.2 Classification Tree Method (CTM)*
CTM [5] is a black-box based testing technique which provides a systematic approach to generate test cases from the functional specification. A *classification-tree* [13] is a finite set of one or more classifications such that: i) there is a specially designated classification A called the root ii) the remaining classifications, which is directly under A are called *subclasses* $A_1$, $A_2$ … $A_n$,

**Table.1 Guidelines for generating equivalence classes for ECPM**

| Valid Input Values | Number of Valid Class | Number of Invalid Class |
|---|---|---|
| Range | 1 (within the range) | 2 (outside the range) |
| Number N | 1 | 2 (none and more than N) |
| Set | 1 (within the set) | 1 (outside the set) |
| Must be | 1 | 1 |

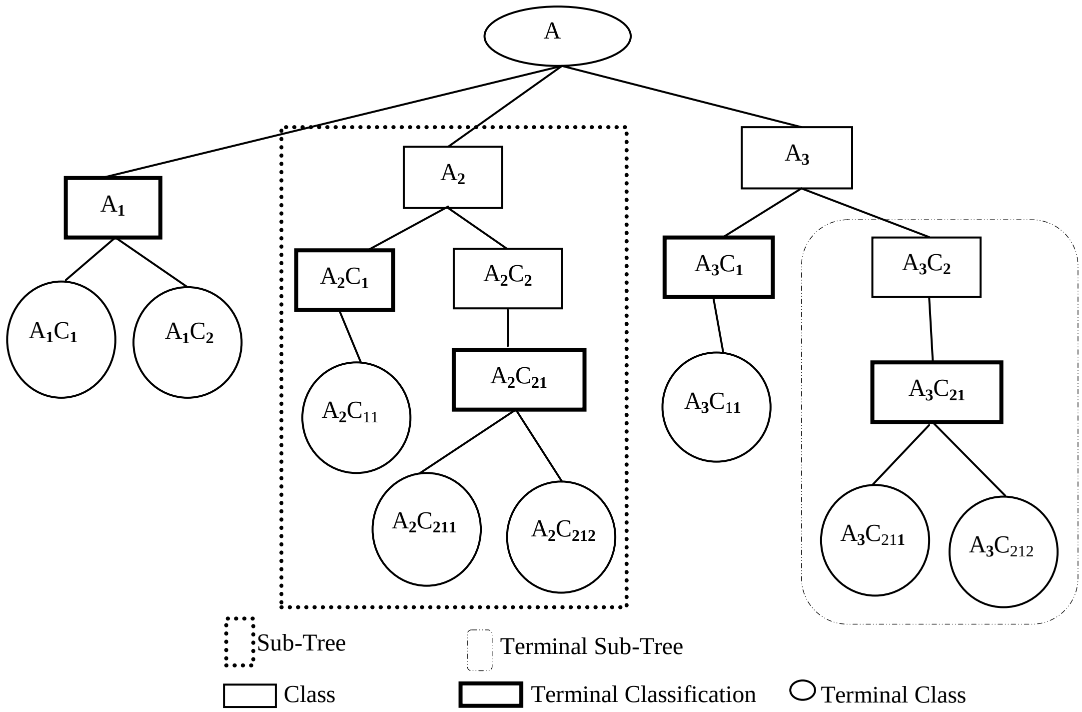


**Figure 1: Example of classification-tree**

where $n \geq 0$ and each of these sets is a tree. $A_1$, $A_2$ … $A_n$ are called the *sub-trees*. Each sub-tree can be subdivided into subclasses. A terminal sub-tree includes a classification with terminal class and their classification. A sub-tree $A_1$ can be represented as: $A1 = \{A_1C_1 \cup A_1C_2 \cup \ldots A_1C_q\}$. Where, $A_1C_1$, $A_1C_2$ … $A_1C_q$ have a common parent classification. $A_1C_j$, $1 \leq j \leq q$ be a sub-tree or terminal sub-tree and q is the number of classifications of $A_1$. Each $A_1C_j$ may be further subdivided into subclasses. In case, if $A_1C_{jm}$, $m \geq 1$ is a terminal classification and $A_1C_{jmp}$, $p \geq 1$ be a terminal class. The corresponding classification tree is shown in Figure.1.

The following algorithm is used to generate classification tree [4]:

1. Construct the sub-trees associated with ancestor relations
2. Rearrange the related sub-trees which is formed in step one
3. Construct sub-trees for standalone classifications
4. Final classification tree is developed by integrating steps 2&3.

We will refer this algorithm as *Construct CT*. In CTM, test cases are generated by combining data values of all different terminal classes. Test selection strategy involves selecting test input from all the terminal classes.

### *2.1.3 Combined Classification Tree Method*

A new test method [8] for Commercial Off-The Shelf (COTS) software based on CTM to identify test cases by first identifying classifications based on the system specifications (in-house) and classifications based on the COTS specifications (external). By overlapping these two classifications a combined classification tree was developed. The combined classification tree is ad-hoc and may vary among testing team. To make this efficient and effective, basic cases for combining two classification trees are presented by Ramadoss and Prema [13].

## 2.2 Test Case Reduction Methods

A good amount of work related to the test case reduction methods is seen in literature. Test suite reduction techniques are broadly discussed by many [6, 9, 14, 15,16, 17]. Also a comparison of test suite reduction is dealt by few [3, 20]. McMaster and Memon [12] have introduced a new coverage criterion for test suite reduction based on the set of unique call stacks. Regression test suite minimization using dynamic interaction patterns with improved fault detection efficiency has been proposed in [19]. Still there exists scope for improvement. A coverage criterion plays a vital role in test case reduction and has been identified by selecting one data value from each class:

(a) *All terminal class coverage* – the selected test set T: $\{tc_1, tc_2, \ldots, tc_n\}$ includes test data from every terminal class, where $tc_1, tc_2, \ldots, tc_n$ are test cases.

(b) *All possible combination of terminal class coverage* – the selected test set T: $\{tc_1, tc_2, \ldots tc_n\}$ includes one of each possible combination of terminal class. Possible combination of terminal class coverage criterion is based on the Cartesian product of the classification. If the terminal classes are inter-dependent, the Cartesian product may generate invalid test cases.

## 3. Home Security System

This section deal with in discussion of test cases generation using ECPM, CTM and combined CTM respectively for a system named Home Security System. Also the factors test case reduction and coverage criteria are discussed.

Home Security System (HSS) is an electronic system that controls the component of an alarm system. It serves primarily to protect a property against intruders by sounding an alarm and / or notifying a central monitoring station if the sensor device detects activity when the system is armed. As such, it provides a valuable line of defense that can protect a home and save lives in the event of an attempted break-in. For the given example, the in-house specifications are listed below:

- Surveillance of the home is 8 to 17 hours
- Alarm system covers only doors and windows
- When the sensor event is recognized, the audible alarm attached to the system.
- The HSS dials a 10-digit phone number to the monitoring service
- It provides information about the location and reporting the nature of the event that has been detected.
- Homeowner receives security information via a control panel, the PC, or a browser, collectively called an interface.

**Table 2. Test conditions for HSS using ECPM**

| Input Partitions | Test Conditions | Valid Input | Invalid Input |
|---|---|---|---|
| Surveillance | 8 to 17 hours | 8 to 17 hours | < 8 hours |
| | | | > 17 hours |
| Alarm System | Doors & Windows | Doors & Windows | ≠Doors &Windows |
| Audible Alarm | ON | ON | ≠ ON |
| Monitoring Service | Any of 10 digit Phone Number | Any of 10 digit Phone Number | ≠ Any of 10 digit Phone Number |

- Interface displays prompting messages and system status information on the control panel, the PC, or the browser window.

### 3.1.1 *ECP Method for HSS*

From the HSS's in-house specification, test conditions for each input are generated using ECP methodology. For example, the 'Surveillance' has one valid test condition '8 to 17 hours' and two invalid test conditions such as '<8 hours' and '>17 hours'. Similarly, 'Audible Alarm' has one valid test condition 'ON' and one invalid test condition '≠ ON'. Table.2 shows the test conditions for HSS.

In HSS, one valid test case is generated for all the valid test conditions and 15 invalid test cases are generated for all the invalid test conditions. Some of these test cases are shown in Table.3. The number of all input coverage criteria for test case generation is obtained based on the total number of invalid test conditions for ECP method. For the given HSS specification, five test cases are needed to cover all input conditions.

**Table 3. Some test cases for HSS using ECPM**

| No. | Test cases | Valid (V) / Invalid (I) |
|---|---|---|
| 1. | {8 to 17 hours, Doors & Windows, ON, 2208759769} | V |
| 2. | {8 to 17 hours, 46127} | I |
| 3. | {Doors & Windows, ≠ ON} | I |
| 4. | {ON, 4 to 7 hours} | I |

### 3.1.2 *CTM for HSS*

For the HSS, the classifications of the in-house specification are 'Surveillance', 'Audible Alarm', and 'Monitoring Service'. The terminal classes are 8 to 17 hours, Doors & Windows, ON, and 'Any 10 digit Phone Number'. Corresponding classification tree is generated using Construct CT algorithm and is shown in Figure.2.

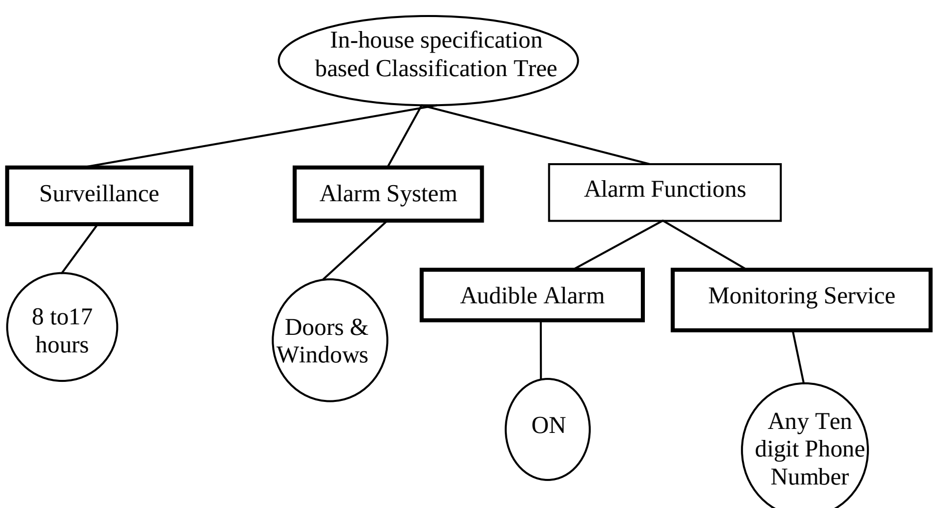


**Figure 2: Classification tree for HSS (In-house specification)**

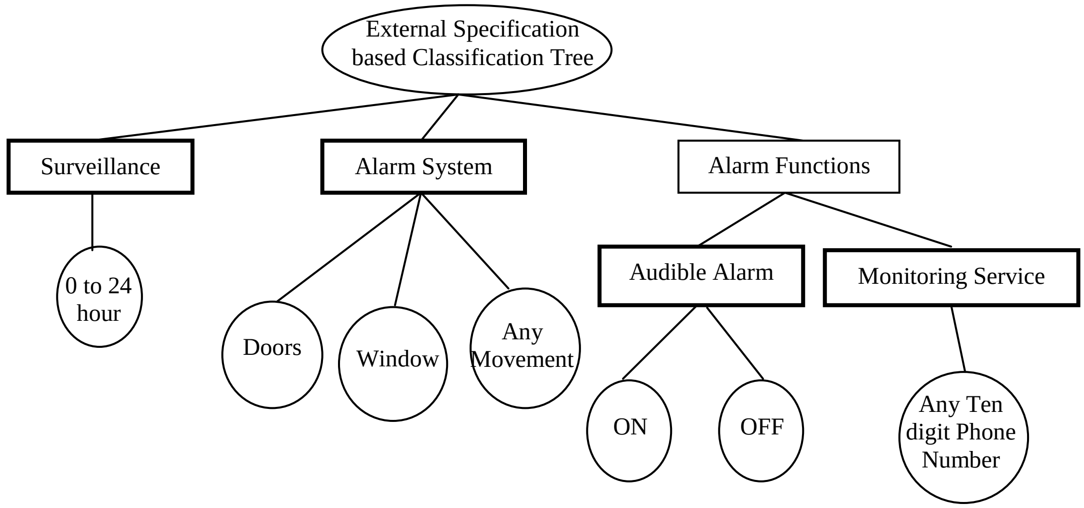


**Figure 3: Classification tree for HSS (External specification)**

By combining one terminal class from each terminal classification test cases are generated. One test case is generated for HSS: {12-15 hours, Door & Window, ON, 2357128791} which also covers all test conditions.

Let the following be the external specification of the HSS:

- HSS offers 24 hours surveillance of the home
- Alarm system covers any of the doors or windows or any movement that takes place in the middle of the night or throughout the day.
- When the sensor event is recognized, the HSS invokes an audible alarm attached to the system.
- The HSS dials a 10 digit telephone number of a monitoring service, provides information about the location, reporting the nature of the event that has been detected.

The classifications of the external specification are: Surveillance, Alarm System, Audible alarm, Monitoring Service. The terminal classes are 0 to 24 hours, Doors, Windows, Any Movement, ON, OFF, Any 10-digit Phone Number. The corresponding classification tree is generated and shown in Figure.3. Six test cases (1*3*2*1) are generated from the classification tree by combining one terminal class from each terminal classification (shown in Table.4). The test cases shown in bold in Table.4 cover all input (here, terminal class) coverage criteria.

**Table.4. Test cases for HSS using CTM (External Specification)**

| No. | Test cases |
|---|---|
| 1. | **{0 to 24, Door, ON, 1342687934}** |
| 2. | {0 to 24, Door, OFF, 2435678542} |
| 3. | **{0 to 24, Window, ON, 2401897657}** |
| 4. | (0 to 24, Window, OFF, 2849582108} |
| 5. | {0 to 24, Any movement, ON, 4393418353} |
| 6. | **{0 to 24, Any movement, OFF, 8832459769}** |

### *3.1.3 Combined classification tree for HSS*

Combined classification tree is generated for HSS based on [8, 13] approaches and using in-house specification based classification tree and external specification based classification tree. The combined classification tree is shown in Figure.4. By combining one terminal class from each terminal classification for the combined classification tree, 18 i.e. 3*3*2*1 test cases are generated (see Table.5). We need only three test cases to cover all input coverage shown in bold in Table.5.

**Table 5. Test cases for HSS using CCTM**

| No. | Test cases |
|---|---|
| 1. | **{8 to 17, Door, ON, 9834589871}** |
| 2. | {10 to 15, Door, OFF, 7873475829} |
| 3. | {8 to 17, Window, ON, 9834589871} |
| 4. | {9 to 11, Window, OFF, 8823445723} |
| 5. | {8 to 10, Any movement, ON, 2208759769} |
| 6. | {16 to 17, Any movement, OFF, 4406175969} |
| 7. | {1 to 6, Door, ON, 7832189871} |
| 8. | {0 to 7, Door, OFF, 9836398731} |
| 9. | **{2 to 3, Window, ON, 1120927341}** |
| 10. | {4 to 6, Window, OFF, 3421942038} |
| 11. | {6 to 8, Any movement, ON, 3203975442} |
| 12. | {2 to 7, Any movement, OFF, 4232978481} |
| 13. | {17 to 24, Door, ON, 8947365831} |
| 14. | {18 to 21, Door, OFF, 5532870839} |
| 15. | {20 to 24, Window, ON, 1182943839} |
| 16. | **{19 to 20, Window, OFF, 2209400881}** |
| 17. | {22 to 24, Any movement, ON, 9804400948} |
| 18. | {17 to 24, Any movement, OFF, 8764900274} |

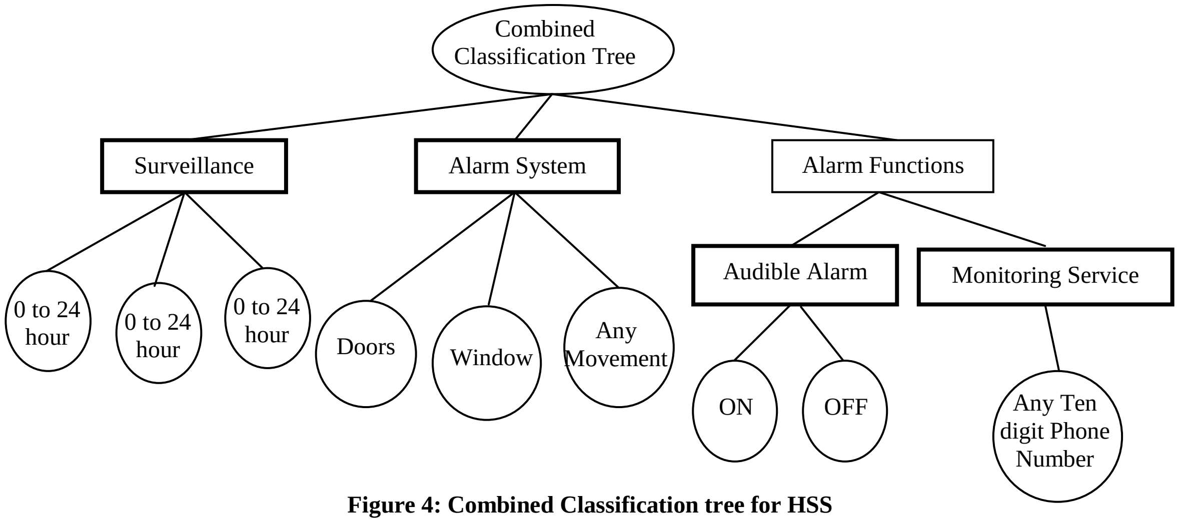


**Figure 4: Combined Classification tree for HSS**

Similarly, these test case generation methods have been applied to several related applications such as Currency Converter, Temperature Controller, Home Security System, Flower Robot, etc.

## 4. COMPARATIVE ANALYSIS

For comparison purpose, case study results specified in earlier section are summarized in Table.6. Various comparisons are made using the parameters: input partition, test conditions, test cases, and coverage criteria. The comparative graph for HSS based on Table.6 is shown in Figure.5, 6 and 7. Results show that, for same number of input partitions, combined classification tree method provides maximum number of test conditions and it yields maximum number of test cases. Also comparing with other methods like ECPM, in-house CTM, and external CTM, combined classification tree method covers all the input conditions with minimum number of test cases. For example, in Figure.7, there are totally 18 test cases for combined classification tree method and only three test cases are needed to cover all input coverage criteria.

Test case generation methods are evaluated based on the percentage of test suite reductions, (to measure the smallest set of test suite). Percentage of test suite reduction of four test case generation methods for HSS is calculated and summarized in Table.7 using the following formula [15, 16]:

**Table.6 Parameters for HSS**

| Y \ X | Number of Input Partitions | Number of Test Conditions | Number of Test Cases | No. of Coverage Criteria |
|---|---|---|---|---|
| In-House ECPM | 4 | 4 | 16 | 5 |
| In-House CTM | 4 | 4 | 1 | 1 |
| External CTM | 4 | 7 | 6 | 3 |
| CCTM | 4 | 9 | 18 | 3 |

Note: X= Parameters, Y = Techniques

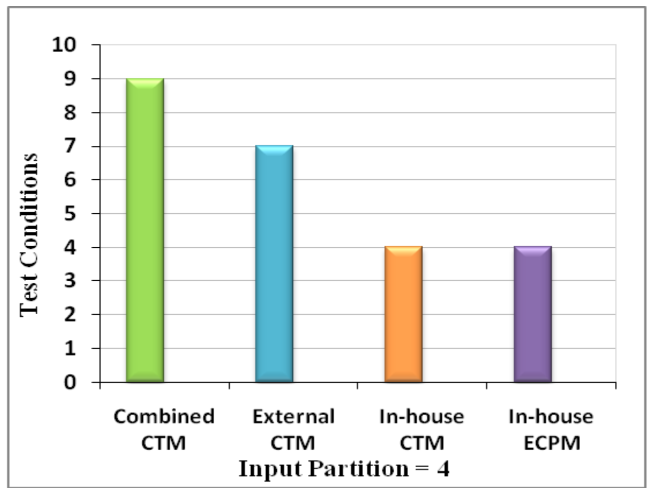


**Figure.5: Input Partition VS Test Conditions**

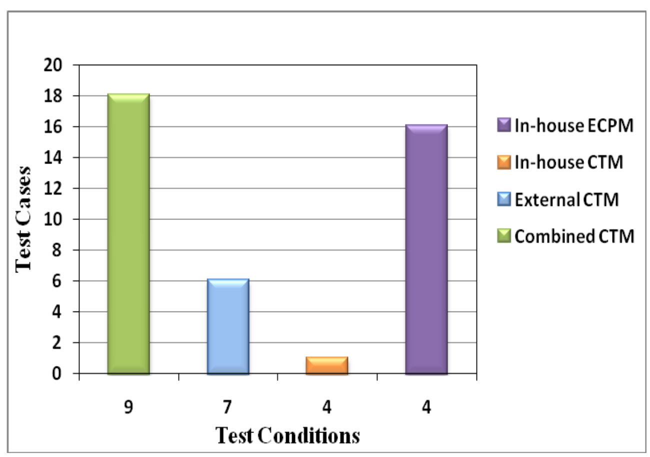


**Figure.6: Test Conditions VS Test Cases**

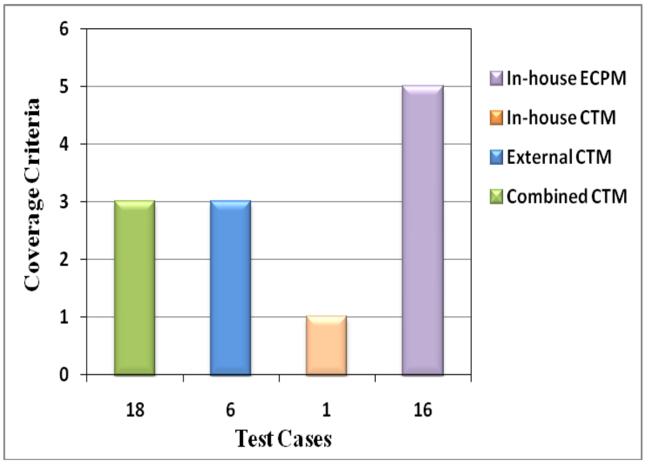


**Figure.7: Test cases VS Test Coverage Criteria**

$$\text{Percentage of test suite reduction} = 100 * \left(1 - \frac{\text{Size}_{\text{Reduced}}}{\text{Size}_{\text{Original}}}\right)$$

The corresponding graph based on Table.7 is shown in Figure.8. The results show that the combined CTM has reduced the test suite by 83.33% (shown in bold) proving its significance over other methods.

**Table. 7 Test Suite Reduction for HSS using Test Case Generation Techniques**

| Techniques | Original test suite | Percentage of reduced test suite |
|---|---|---|
| In-house ECPM | 16 | 68.75% |
| In-house CTM | 1 | 0% |
| External CTM | 6 | 50% |
| Combined CTM | **18** | **83.33%** |

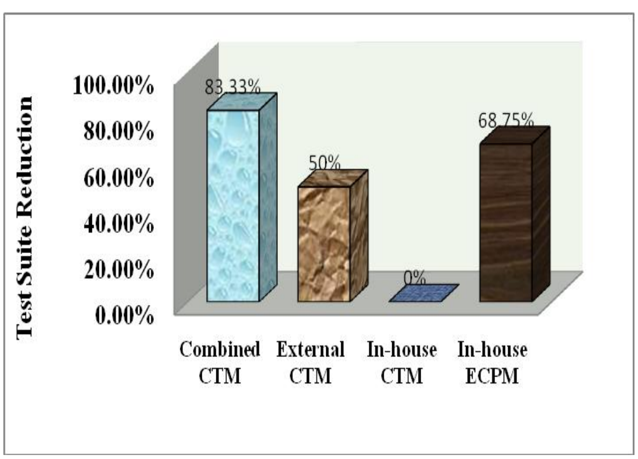


**Figure 8: Comparison of Test Suite reduction for HSS**

## 5. CONCLUSION

Test suite reduction techniques are used in this paper to reduce the number of test cases with respect to some coverage criteria. Partition based test case generation techniques such as in-house ECP method, in-house and external CTM's and combined CTM are compared in this paper. It highlights the significance of CTM and CCTM over ECPM for test case generation. It is observed that CCTM provides a better test suite reduction for all input coverage criteria.

## 6. REFERENCES


[1] Ammann, P. E., and Black, P. E. 2001. A specification-based coverage metric to evaluate test sets. *International Journal of Reliability, Quality and Safety Engineering*. 8, 4 (Dec. 2001), 275-300.

[2] Binder. R. V. 1996. Testing object-oriented software: a survey. *Software Testing Verification and Reliability*, 6, 3/4, 125 – 252.

[3] Chen, T. Y., and Lau, M. F. 1998. A simulation study on some heuristics for test suite reduction. *Information and Software Technology*. 40, 13, 777-787.

[4] Chen,T.Y., and Poon, P.L.1997. Construction of classification trees via the classification-hierarchical table. *Information and Software Technology*. 39, 13, 889-896.

[5] Grochtmann. M., and Grimm. K. 1993. Classification trees for partition testing. *Verification and Reliability*,3, 2, 63-82.

[6] Harrold, M. J., Gupta, R., and Soffa, M. L. 1993. A methodology for controlling the size of a test suite. *ACM Transactionson Software Engineering and Methodology (TOSEM)2, 3*(July 1993).

[7] Hennessy, M., and Power, J. F. 2005. An analysis of rule coverage as a criterion in generating minimal test suites for grammar based software. *In Proceedings of the 20th IEEE/ACM International Conference onAutomated Software Engineering (ASE'05)*, (Long Beach, CA, USA, Nov. 2005), 104–113.

[8] Leung, H., and Prema, P. 2003. Testing COTS with classification tree method. In *Proceedings of the IASTED International Conference on Software Engineering and Applications* (L.A., Nov. 2003). SEA 2003, 270-276.

[9] Kichigin, D. 2010. A Method for Test Suite Reduction for Regression Testing of Interactions between Software Modules. *LNCS, Springer – verlag*, 2010, 177-184.

[10] Mathur. A. P. 2008. Foundations of Software Testing: Fundamental Algorithms and Techniques. Pearson Education.

[11] Myers. G. J. 1979. The Art of Software Testing. John Wiley & Sons, New York.

[12] McMaster, S., and Memon, A.2005. Call stacks coverage for test suite reduction. *In Proceedings of 21st IEEE International Conference on Software Maintenance.539-548.*

[13] Ramadoss, B., and Prema, P. 2009. An approach for merging two classification trees. In *Proceedings of the IEEE International Advance Computing Conference* (Patiala, Punjab), 1829 – 1834.

[14] Rothermel, G., Harrold, M. J., Ostrin, J. and Hong, C. 1998. An empirical study of the effects of minimization on the fault detection capabilities of test suites. *Proceedings of the International Conference on Software Maintenance*, (Nov. 1998) 34-43.

[15] Rothermel, G., Harrold, M. J., Von Ronne, J., and Hong, C. 2002. Empirical studies of test-suite reduction. *Journal of Software Testing, Verification and Reliability*. 12, 4. DOI = 10.1002/stvr.256.

[16] Wong, W. E., Horgan, J. R., Mathur, A. P., and Pasquini, A. 1997. Test set size minimization and fault detection effectiveness: A case study in a space application. *Proceedings of the 21st Annual International Computer Software and Applications Conference.* 522-528.

[17] Wong, W. E., Horgan, J. R., London, S., Mathur, A. P. 1995. Effect of test set minimization on fault detection effectiveness. *Proceedings of the 17th InternationalConference on Software Engineering*. (Washington, 1995), 41-50.

[18] Samuel, P., and Mall, R. ( 2008). A novel test case design technique using dynamic slicing of UML sequence diagrams. *E-Informatica Software Engineering Journal*. 2, 1 (2008).

[19] Selva Kumar, S., and N. Ramarai. (2011). Regression test suite minimization using dynamic interaction patterns with improved FDE. *European Journal of Scientific Research*, 49, 3 (2011), 332-353.

[20] Zhong, H., Zhang. L., and Mei, H. 2006. An experimental comparison of four test suite reduction techniques, ICSE 2006, ACM, 636-639.