Enhancing Acceptance Test Driven Development Model with Combinatorial Logic

Subhash Tatale[1]
Research Scholar, Dept. of CSE
Koneru Lakshmaiah Education Foundation
Vaddeswaram Vijayawada, India

Dr. V. Chandra Prakash[2]
Professor, Dept. of CSE
Koneru Lakshmaiah Education Foundation
Vaddeswaram Vijayawada, India

***Abstract*—In the Software Development Life Cycle, modelling plays a most significant role in designing and developing software efficiently. Acceptance Test-Driven Development (ATDD) is a powerful agile software development model where a customer provides user acceptance test suits as a part of Software Requirements Specifications. A design has to develop a system so that User Acceptance Tests will be successful. In some systems, the Combinatorial Logic and Combinatorial Testing play a very crucial role. The authors have proposed a novel approach to enhance the existing Acceptance Test Driven Development model to Combinatorial Logic Oriented-ATDD model by incorporating combinatorial logic. Refinement with respect to combinatorial logic needs to be incorporated in all the stages of Software Development Life Cycle, i.e. starting from Software Requirement Specifications to User Acceptance Tests. This comprehensive approach derives the acceptance tests from user requirements effectively and efficiently. In this paper, the existing Indian Railway Reservation System is considered as a case study, and it was fully implemented as per proposed Combinatorial Logic Oriented-ATDD model.**

Keywords*—Software requirements specification; software development life cycle; acceptance test driven development; combinatorial logic; combinatorial testing; user acceptance tests; railway reservation system

I. Introduction

This Nowadays, software systems are becoming increasingly complex. It is more challenging to verify the correctness of complex software requirements specification [1]. Formal verification approaches are highly sensitive to the software's complexity and might require costly resources, namely tools and human resources. During functional testing, many errors can be captured. It is not very easy to validate software requirements specification from a potentially huge set of parameters, values, or conditions of the system [2][3].

In the systems such as reservation system, college admission system, concession management system, etc. combinations of a set of parameters, values, or conditions are present. Combinatorial logic plays a considerable role in such type of systems. For example, in the current railway reservation system, a passenger avail only one type of concession at a time. A passenger can avail multiple concessions at a time by applying combinatorial logic on a set of concession categories and types. Suppose a passenger who wants to avail concession is a senior citizen and physically handicapped, then the passenger can get more percentage of concession than percentage of concession offered in single concession using pairwise combinatorial logic. If a passenger is a senior citizen, cancer patient, and physically handicapped, then the passenger can get more percentage of concession than percentage of concession offered in pairwise concession using 3-way combinatorial logic. In addition to standard testing techniques, combinatorial testing is also very much essential to test this combinatorial logic to ensure the reliability of such systems.

The combinatorial logic is applied to various systems and performed combinatorial testing to ensure the reliability of those systems. The researchers claimed that 67% of the faults were triggered by only a single parameter value, 93% by 2-way combinations, and 98% by 3-way combinations of a complex application. For some applications, 100% faults were detected with 4 to 6- way interactions [4].

The authors made a survey to find out a model among the classical process models viz. waterfall, spiral, incremental, etc., and agile process models viz. extreme programming, scrum, etc., that is most suitable to represent combinatorial logic. The authors found that Test Driven Development (TDD) and Acceptance Test Driven Development (ATDD) are more suitable to represent combinatorial logic.

There are many systems where combinatorial logic is found as a must and hence there is need to put focus on how to incorporate combinatorial logic in all the stages of Software Development Life Cycle (SDLC) in a better way. Combinatorial logic can be incorporated into all the stages of SDLC. Out of earlier mentioned process models, the authors identified that the TDD and ATDD models are suitable to represent the combinatorial logic. In this section, concepts related to combinatorial testing and the ATDD model are discussed.

A. Combinatorial Testing

Combinatorial Testing (CT) is a specification based technique. It provides a systematic way to select combinations of program inputs or parameters for testing. It is a useful testing technique to test hardware or software system which identifies failures based on input or output combinations of parameters. Over the years, this technique has been applied to test system configurations, web forms, protocols, graphical user interfaces, and software product lines [5]. The possible n-way (n=2, 3, 4, 5, 6, or more) combinatorial interactions

among the input variables can be detected by testers using the combinatorial testing technique [6].

1) Pairwise testing: Pairwise testing technique is the most commonly used combinatorial testing. It is a useful testing technique which involves all possible discrete combinations of each pair of input parameters of the system. Pairwise testing can be performed much faster than exhaustive testing that tests all combinations of all input parameters. The majority of software faults are triggered by a single input parameter or a combination of two input parameters. Pairwise testing requires that each pair of input parameter values should be represented at least by one test case. Let us consider software that takes three input parameters say x, y, z. If each parameter has four different values, then there will be 64 different pairs; {(x1, y1) (x1, y2) ……(y4, z4)}. A test case (x1, y3, z2), for example, represents three of these 64 pairs: (x1, y3), (x1, z2), (y3, z2). The pairwise testing method is highly useful for cases with a limited number of parameters with multiple possible values to reduce the test suite size and detect about 70% to more than 90% of software faults [7].

2) n-way testing: Some faults are triggered only by a combination of 3, 4, or more parameter values. These combinations cannot be detected by the pairwise testing. There is a need to test 3-way and 4-way combinations of parameter values for those cases. The study showed that across a variety of domains, all faults are detected by a maximum of 4-way to 6-way interactions. The fault detection rate increases rapidly with interaction strength (n-way combinations).

B. Acceptance Test Driven Development

Acceptance Test-Driven Development (ATDD) [8] supports collaboration among the customers, developers, and testers to ensure that acceptance tests exist before writing any code. The acceptance tests are written from the perspective of the end-user. In the ATDD model, acceptance tests are written before developers start coding. The ATDD model has been used from time to time by considering the following goals [9].

1) ATDD is a specification and not validation. It is one way of thinking through the software requirements specifications followed by user acceptance tests before writing the functional code.

2) ATDD is simply a programming technique to write a clean code that works effectively.

3) ATDD is not testing software, but it stands as an aid to the programmer and customer during the development process to establish unambiguous requirements.

There is a scope for research to apply the ATDD model to the applications where requirements are specified using the combinatorial logic. This paper is organized as follows. Section 2 reviews the related work on combinatorial testing and the ATDD model. Section 3 discusses the classical and combinatorial logic oriented -ATDD model. In section 4, Railway Reservation System based on combinatorial logic oriented-ATDD model is presented. Section 5 reports experimental results. Section 6 concludes this paper and provides the guidelines of the future work.

II. Related Work

Several researchers have attempted to pursue research in the field of combinatorial testing technique and ATDD model. In this section, related work on combinatorial testing and the ATDD model has been discussed.

A. Combinatorial Testing

The combinatorial testing technique used to generate tests that cover pairwise or n-way combinations of parameters of the system by implementing the AETG system. The AETG system [10] is in a variety of applications for the unit, system, and interoperability testing. Automated Combinatorial Testing for Software Tool (ACTS) developed to apply high-strength combinatorial testing to detect intangible failures that occur when multiple components interact with each other. ACTS tool uses IPOG, IPOG-F, IPOG-F2, IPOG-D, and Base Choice algorithms for test case generation [11-13]. The fault localization approach [14] is used that can help programmers in locating faults with less manual interference. The available algorithms/tools of combinatorial testing are categorized based on different comparison criteria [15], including the test suite generation technique, combination criteria, mixed covering array, the strength of coverage, and the support for constraints between parameters.

The different search algorithms like the Genetic Algorithm, Particle Swarm Optimization, Ant Colony Algorithm, Bee Colony Optimization, and Simulate Annealing [16-17] are used to test embedded systems using test cases generated through combinatorial testing techniques. The combinatorial test cases are derived from the output domain in systems such as safety-critical embedded systems, which ensure maximum output combinations tested in detail. These test cases are derived using a genetic algorithm [18-19] and adjacent pairwise testing [20]. The various constraint handling, identification, and maintenance techniques [21] of combinatorial testing are analyzed. A Neural Network approach [22] is used to improve combinatorial coverage in the combinatorial testing approach [23]. A multi-objective crow search and fruit-fly optimization techniques [24] are used to optimize combinatorial test cases in constraints handling environment.

System requirements and corresponding models [25] are proposed for applying the combinatorial approach to those requirements. A structured modelling method [26] used to translate requirements expressed in a general format into an input parameter model suitable for combination strategies. A number of Articles [27-39] have been presented for testing embedded systems using combinatorial methods in the literature for testing distributed embedded systems.

B. Acceptance Test Driven Development Model

ATDD is the developer-focused model where the entire team collaborates to define the acceptance criteria of a user scenario before the actual implementation starts. ATDD model is implemented using the Given-When-Then format [40] that uses unit tests to deliver small pieces of functionality

incrementally. A hybrid approach of combining user-centered agile methodology with ATDD model [41] is proposed in an efficient manner. It makes the possibility of software reusability based on the needs of end-users for decreasing development costs. The various case studies of a Real-Time Embedded system [42-43] and web applications [44] from industry are developed using Project-Based Learning [45], ATDD, and agile development. This helps to detect unauthorized access and fraud.

A combination of ATDD and Model-Based Testing (MBT) [46] is applied in several real-world projects. This approach increased test coverage and extended testing to user scenarios. It is exercised by the existing acceptance tests to minimize the risks and to reduce the effort involved in introducing MBT in the projects. The idea of collaborating Quality Function Deployment (QFD) and ATDD [47] is proposed. The principles of QFD are applied to capture the customer requirements and deploy them into functional and non-functional requirements. These requirements are mapped into user scenarios, which then became the acceptance tests. The development is performed based on those acceptance tests using the ATDD model. Production code is validated later against the customer requirements instead of the interpretation of the requirements by the developer team. An AnnoTest Web/Run tool [48] is used by expert customers to specify acceptance tests through the reuse of existing requirements specification.

The development teams have a better understanding of the software requirements as it mandates the exact behavior in terms of acceptance criteria using ATDD. The improved understanding of requirements results in reduced defect density and hence reduced Cost of Quality. This improvement also helps in simplifying the need for repetitive or breakthrough improvement as per changing business requirements. The ATDD model is an effective way of developing an application in a continuously evolving environment [49]. The open-source Quality Assurance of Complex Event Processing (CEP) Testing System [50] is introduced for realizing the executable acceptance test-driven development of complex event processing applications.

III. Combinatorial Logic Oriented-ATDD (CLO-ATDD) Model

Combinatorial logic plays an essential role in designing and developing systems like Reservation Systems, Concession Management System, College Admission System, etc.

As mentioned in section I, the authors found that TDD and ATDD models are suitable to represent combinatorial logic. In the TDD model, test cases are written in the same language in which the features are implemented. If the features are implemented in Java programming language, then test cases are written in Java (e.g., JUnit test cases which are written Java). The TDD model focuses on the implementation of the features. In the ATDD model, test cases are written in simple business language. The authors propose that combinatorial logic can be incorporated in acceptance tests of the ATDD model. User acceptance tests are written from the user's point of view. Developers implement the system using these user acceptance tests. Hence, ATDD is more suitable model than TDD to incorporate combinatorial logic while framing the SRS document. The authors propose enhancement in the existing ATDD model by incorporating combinatorial logic in all the stages of SDLC. In the next section, classical SDLC and Combinatorial Logic Oriented-ATDD (CLO-ATDD) models are explained.

A. Classical SDLC

SDLC defines a methodology for the overall development process, which improves the quality of software. It consists of a detailed plan illustrating how to develop, maintain, replace, and enhance specific software.

The following are the various stages of a classical SDLC.

1) Communication.
2) Planning.
3) Modeling.
4) Construction.
5) Deployment.

Every software process has its limitations, and the SDLC stands as unexceptional to that. The selection of the appropriate SDLC model is a very challenging task. Each model has definite advantages and disadvantages; therefore, it is essential to assess each one to ensure fitness. Most SDLC models are designed around a business partner or customer requirements. It is difficult for business partners and customers to deliver the detailed requirements specification of the systems, which is to be developed as per their expectations within time, cost, and quality. A successful implementation requires dedicated user involvement to capture the true essence of the system requirements. If the business partners or customers are not satisfied with the working functionality/features, the development team has to modify the functionality/features. Multiple modifications in software development cause a potential delay in deliverable components. If changes are delayed to be implemented in the process, it increases the total cost of the system while extending the time to completion. In the next section, the authors proposed CLO-ATDD model.

B. Proposed CLO-ATDD Model

The proposed CLO-ATDD model is an enhancement of the existing ATDD model. In CLO-ATDD, user acceptance tests are prepared in a business language. These tests are prepared based on the combinatorial logic oriented rules, as discussed in section 3.2.1. Gherkin syntax [51] is used to prepare the test cases. It is easy to learn Gherkin syntax which is specified in a business language. The Gherkin syntax has a structured format to illustrate the business rules of real-world applications. The user acceptance tests are prepared using a well-defined Given-When-Then structure format and a few keywords. In Gherkin syntax, each feature contains multiple user scenarios. Each feature starts with a keyword, followed by a description of the feature. Each test starts with a sequence of these keywords: Given, When, Then. And and But is used whenever necessary. The description of these keywords is given below.

- Given – It describes the preconditions for the scenario.

- When – It describes the operation that we want to test.
- Then – It describes the expected result.
- And and But – They are optional. These keywords are used as conjunctions and semantically continue the meaning of previous sentences.
- Comment –It is optional. This can be used to provide explanation of the test case.

In this section, the CLO-ATDD model is discussed. The different stages of SDLC with respect to the CLO-ATDD model are described below.

1) Communication: In the classical SDLC, only requirements are prepared and finalized during this phase. In ATDD, communication among the business customers, the developers, and the testers happen to discuss the requirements specification. In ATDD, acceptance tests are written before the developer team starts coding. In CLO-ATDD model, the following activities are carried out.

a) Preparation & finalization of the SRS document of the system by different stakeholders such as developer, customer, etc.

b) This SRS document is prepared using combinatorial logic oriented rules. The rules are an essential part of the SRS document. The SRS consists of a set of scenarios, configurations, or conditions. The combinatorial logic is applied to these sets of scenarios, configurations, or conditions to prepare the combinatorial logic oriented rules. These combinatorial logic oriented rules are used by a business analyst to analyze the system, a designer to design the system, by a programmer during coding, and by a tester to test the system using combinatorial testing.

c) Preparation & finalization of the test suite for the user acceptance tests. Test cases are prepared from the requirements specification for the user acceptance test. These user acceptance tests are prepared by using the Given-When-Then format [51].

The SRS document is very much important in this phase of the CLO-ATDD model. It consists of combinatorial logic oriented rules and user acceptance tests.

2) Planning: The planning phase consists of project cost estimation, project scheduling, and resources like human resources, hardware, software, and network resources. The team members are allocated as per skill sets of the members for the project's active development. In the proposed approach, combinatorial testing is very much essential. The team members having skill-sets of combinatorial testing are preferred in this model. Regarding software resources, many combinatorial logic-oriented tools are available. Developing new tools require more time and cost. Existing available combinatorial testing tools like AETG and ACTS are used to complete the projects as per the schedule and to save the project's cost.

3) Modeling: In this section, the analysis and design of the proposed model are discussed. Combinatorial logic is applied while designing the system.

In the analysis stage, an in-depth analysis of the requirements is performed to obtain a detailed understanding of the system's business needs. System requirements are studied and structured. The result after this stage is a requirement document called the SRS document. The SRS document tries to capture the requirements from the customer's perspective. Combinatorial logic oriented UML diagrams are drawn to specify, analyze, and visualize the requirements specification.

It is an important stage in which the requirements specifications are designed by using combinatorial logic. This design is represented by a set of parameters, their respective values, and constraints on the value combinations. If a set of parameters and their respective values are high, then a huge number of parameter-value combinations are generated.

There is a need to give solutions for efficiently generating input combinations to represent software interaction and generation of test suites using efficient techniques. While generating test cases automatically for conducting combinatorial testing, sometimes, the size of test suite may be extremely large because of too many parameters and values in input. This is called as Combinatorial Explosion of test cases. To avoid the occurrence of the combinatorial explosion, we proposed CLO-ATDD design.

4) Construction: In this phase, the system is implemented and tested by using combinatorial logic. Many algorithms are available for generating combinatorial test suites.

After completion of the system designing phase, the coding phase begins. In this phase, developers start to develop the system by writing code based on the combinatorial logic oriented rules. It is the most extended phase of the SDLC process. The stakeholders should be involved regularly to ensure that their expectations are being fulfilled. The output of this phase is testable and functional software.

Exhaustive testing of any system may be impossible sometimes because the domain of input parameters to most software systems is huge. There is a need to design optimized test suites that are of reasonable size and are useful to detect as many defects present in the system as possible. If test cases are selected randomly, many of these randomly selected test cases do not contribute to the significance of the test suite. Thus, the number of random test cases in a test suite is not an indication of the effectiveness of the testing. User acceptance tests are prepared based on the combinatorial logic oriented rules. User acceptance testing is defined as testing the software by the user or client to verify whether it can be accepted or not.

5) Deployment: Once the software testing phase is completed, and defects are not present in the system, then the final deployment process starts. As per the proposed CLO-ATDD model, the final software is deployed after the user

accepts the user acceptance tests. The deployment manager does the deployment.

IV. Railway Reservation System: A Case Study of the Proposed Model

In this section, a case study of Concession Management Subsystem (CMSS) of the proposed Railway Reservation System (RRS) is explained by using CLO-ATDD model.

A. Framing SRS using CLO-ATDD Model

Indian Railways is one of the most prominent organizations of the Government of India. Indian Railways offers concessions on ticket fares with respect to different concession categories such as Disabled Passengers, Patients, Senior Citizens, Awardees, War Widows, Students, and Youths etc. These concessions are available for various types of journey classes viz. Sleeper (SL) Class, Second Class (2ND), First Class (1ST), 1-2-3- Tier AC Class and Ac Chair. The list of different concession categories and concession types along with the journey class are shown in Table I.

1) Limitations of the existing reservation policy of Indian Railways:

a) All the concession types mentioned in Table I are not available on on-line ticketing platform of Indian Railways. The concession types like senior citizens, divyangjan, general, press person, press child, press spouse, and press companion are available on the on-line ticketing platform. The remaining concession types can be availed by the passenger at the Passenger Reservation System (PRS) counters at any reservation office of Indian Railways.

b) Only one type of concession is applicable at a time as per the choice of a passenger. The passenger is not allowed to avail more than one concession at a time.

There is a need to re-design, re-develop the railway reservation system to overcome the limitations of the existing railway reservation system. In the next section, the authors proposed a new reservation system by revising the software requirements specification of the existing Railway Reservation System by using the CLO-ATTD model.

2) Proposal for revision of SRS using CLO-ATDD Model: The following are subsystems of the present RRS of Indian Railways which provide on-line facility for reservation.

- Search Train.
- Plan Journey.
- Cancel Ticket.

The authors propose a new RRS by incorporating combinatorial logic in the system using CLO-ATDD model. In the new RRS, Concession Management Subsystem (CMSS) is enhanced to manage the concessions. CMSS is used to manage multiple concessions at a time which can be availed by a passenger. Here, a novel approach is proposed to enhance the CMSS so that a passenger can be benefited by availing multiple concessions at a time. The existing SRS document is enhanced by incorporating combinatorial logic oriented rules and user acceptance tests. In the SRS document, two sections are added. These sections are very much important. First section consists of combinatorial logic oriented rules and second section consists of user acceptance tests. The proposed rules for availing multiple types of concession are shown in Table II. User acceptance tests are prepared using the Given-When-Then format as shown in Table III.

B. Design of Concession Management Subsystem

Software systems have five types of design such as Database Design, Program Architecture Design, File Design, Input Design and Output Design. Screen design can be used to represent both input and output design of a system wherever it is relevant. Now-a-days, Graphical User Interface (GUI) is used to design screens. In this section, the authors propose combinatorial logic oriented design of CMSS of RRS.

In the existing RRS, a passenger has to fill up a concession form at passenger reservation counter at any railway reservation office. There are some concessions viz. senior citizen, child, divyangjan etc. for which there is no need to fill up a separate concession claim form. The proof of these concession types is verified by a reservation clerk. For other concession types, the passenger has to fill up a concession claim form and submit it along with the necessary proofs to the reservation manager. The manager verifies the document proofs for the claim and sanctions concession for only one concession type as mentioned in Table I. The reservation manager issues a document termed as concession sanction order which consists of the maximum percentage of concession sanctioned along with signature and stamp of reservation manager. All this procedure is to be performed off-line and a lot of time may be required to complete this procedure. After this, the passenger will go to reservation counter to reserve his/her ticket. The reservation clerk verifies the concession sanction order, provides concession in the ticket fare, and reserves the seat/berth for the passenger.

The authors proposed a new CMSS of RRS using combinatorial logic so that a passenger can avail more than one concession type at a time. A passenger uses CMSS and fills up a concession claim form by selecting one or multiple concession types and submits it along with the necessary proofs to reservation manager. The manager will verify all the concessions claims and sanctions the total concession allowed to the passenger by using CMSS. The reservation manager uses CMSS and issues the concession sanction order consist of total percentage of concession offered along with signature and stamp. The passenger will reserve the ticket as per the total percentage of concession sanctioned.

TABLE I. LIST OF CONCESSION CATEGORIES AND CONCESSION TYPES ALONG WITH JOURNEY CLASS

Category of Concession	Journey Class						
	SL	2ND	1ST	1AC	2AC	3AC	AC Chair
	% of Concession						
Disabled Passengers							
Orthopedically Handicapped, Mentally retarded, Blind	75	75	75	50	50	75	75
Deaf & Dumb	50	50	50	NA	NA	NA	NA
Patients							
Cancer	100	75	75	50	50	100	75
Thalassemia, Heart, Kidney	75	75	75	50	50	75	75
Hemophilia	75	75	75	NA	NA	75	75
T.B./Lupus Valgaris, Non-infectious Leprosy	75	75	75	NA	NA	NA	NA
AIDS	NA	50	NA	NA	NA	NA	NA
Sickle & Aplastic Anaemia	50	NA	NA	NA	50	50	50
Senior Citizens							
Men- 60 years and above.	40	40	40	40	40	40	40
Women- 58 years and above	50	50	50	50	50	50	50
Awardees							
President's Police Medal, Indian Police Award (Age>=60)	50M/ 60F	50M/60F	50M/60F	50M/60F	50M/60F	50M/60F	50M/ 60F
Shram	75	75	NA	NA	NA	NA	NA
National Awardee Teachers /Bravery Award	50	50	NA	NA	NA	NA	NA
War Widows							
War Widows, Widows[3]	75	75	NA	NA	NA	NA	NA
Students							
SC/ST Category for hometown & educational tours	75	75	NA	NA	NA	NA	NA
Students of Govt. schools for study tour, Entrance exam - Girls of Govt. schools in rural areas	NA	75	NA	NA	NA	NA	NA
Main written examination conducted by UPSC & SSC	NA	50	NA	NA	NA	NA	NA
Foreign students, Research scholars' for journeys[4], Cadets and Marine Engineers[5] , , Hometown & educational tours	50	50	NA	NA	NA	NA	NA
Students and non-students participating in Camps	25	25	NA	NA	NA	NA	NA
Youths							
National Youth Project, To attend job interview in Statutory Bodies, Bharat Scouts &	50	50	NA	NA	NA	NA	NA
ManavUththanSewaSamiti	40	40	NA	NA	NA	NA	NA
To attend job interviews in Central & State Govt.	50	100	NA	NA	NA	NA	NA
Kisans							
Kisans and Industrial Labourers	25	25	NA	NA	NA	NA	NA
Kisans travelling	33	33	NA	NA	NA	NA	NA
Kisans& Milk Producers, Delegates for attending Annual Conferences[6]	50	50	NA	NA	NA	NA	NA
Artists & Sportspersons							
Artistes & Film technicians	75	75	50	NA	50	50	50
All India, State, National tournaments Sportsmen, Mountaineering Expeditions	75	75	50	NA	NA	NA	NA
Press Correspondents	50	50	50	50	50	50	50
Medical Professionals							
Doctors – Allopathic - travelling for any purpose	10	10	10	10	10	10	10
Nurses & Midwives - for leave and duty	25	25	NA	NA	NA	NA	NA
(3 widow of I.P.K.F. Personnel, Policemen & Paramilitary personnel, defense personnel, Martyrs of Operation Vijay in Kargil, 4 journey in connection with research work. (age<=35), 5 apprentices for travel between home and training ship, 6 delegates of Bharat Krishak Samaj & Sarvodaya Samaj, Wardha)							

TABLE II. RULES FOR AVAILING CONCESSION IN SRS DOCUMENT USING CLO-ATDD MODEL

Rule No.	Criteria	% of total concession
1	No. of concession types selected= 1	% of total concession is applicable as per Table I.
2	No. of concession types selected= 2	% of total concession = % of highest concession type + 5% of remaining concession type
3	No. of concession types selected= 3	% of total concession = % of highest concession type + 7% of remaining higher concession type
4	No. of concession types selected > 3	% of total concession = % of highest concession type + 10% of highest of the remaining concession type
5	% of total concession exceeds maximum allowed concession	% of total concession = maximum allowed concession (=100%)

TABLE III. SAMPLE USER ACCEPTANCE TEST

Test Case No.	Keyword	Description
1	Given	The passenger selects concession type as orthopedically handicapped for second class journey.
	When	Rule 1 is applied. (Refer Table II)
	Then	The passenger gets 75% concession in ticket fare. (Refer Table I)
2	Given	The passenger selects concession types as an orthopedically handicapped and a cancer patient for second class journey.
	When	Rule 2 is applied.
	Then	The passenger gets 78.75% concession in ticket fare.
	Comment	% of total concession= 75% (orthopedically handicapped) + 5% of 75 (cancer patients) = 75 % + 3.75% = 78.75% (Refer Table I & Rule 2 of Table II)
3	Given	The passenger selects concession types as a female senior citizen, an orthopedically handicapped and a cancer patient for second class journey.
	When	Rule 3 is applied.
	Then	The passenger gets 80.25% concession in ticket fare.
	Comment	% of total concession= 75% (orthopedically handicapped) + 7% of 75 (cancer patients) = 75 % + 5.25% = 80.25% (Refer Table I & Rule 3 of Table II)
4	Given	The passenger selects concession types as a female senior citizen, war widow, an orthopedically handicapped and cancer patient for sleeper class journey.
	When	Rule 4 & Rule 5 are applied.
	Then	The passenger gets 100% concession in ticket fare. (Refer Table I & Rule 4 of Table II)
	Comment	% of total concession= 100% (cancer patient) + 10% of 75 (war widow) = 100 % + 7.5% = 107.5% (As per rule 4) As per rule 5, % of total concession = 100%

For input and output design, GUI based screen is designed as shown in Fig. 1. The CMSS is role based viz. passenger and reservation manager. The GUI consists of all concession categories and types as mentioned in Table I. A passenger has to fill up his/her personal information, journey details, and has to select one or multiple concession types. For example, the passenger can select concession types as a female senior citizen, war widow, an orthopedically handicapped and cancer patient for sleeper class journey and submits the form. A passenger cannot select infeasible combination of concession types. For example, a male senior citizen passenger cannot select for concession type of war widow. The GUI is designed in such as way that a passenger cannot select infeasible combination of concessions types. When a passenger is filling the concession claim form, only Submit and Print buttons are enabled and Approve button is disabled.

In case of a reservation manager, Approve and Print buttons are enabled. The reservation manager verifies the document proofs of selected concessions claimed by the passenger. If any claim is not valid then the manager can unselects the concession claim. The manager can then approve the concession claim. The combinatorial logic oriented rules mentioned in Table II are used by CMSS to calculate total concession in ticket fare. Finally, the concession sanction order is printed which consists of original ticket fare, total number of concessions submitted by the passenger, total number of concessions sanctioned by the reservation manager, and net ticket fare.

C. Implementation

In this section, implementation strategy of proposed CMSS and technique of test case generation is discussed.

1) Implementation strategy of proposed CMSS: The inputs of the proposed system are parameters and values. In CMSS, the concession categories and concession types are considered as parameters and values, respectively. Each concession category includes multiple concession types. The CMSS system has some constraints and conditions; for example, a male passenger cannot avail concessions of widow concession type because of infeasible combinations of parameters and values.

Client-server architecture is used to implement the proposed CMSS, as shown in Fig. 2. Interactive web pages at the client-side are designed by using Vue JS open source JavaScript framework. Express Node.js framework is used to develop a robust set of features of the proposed system at the server-side.

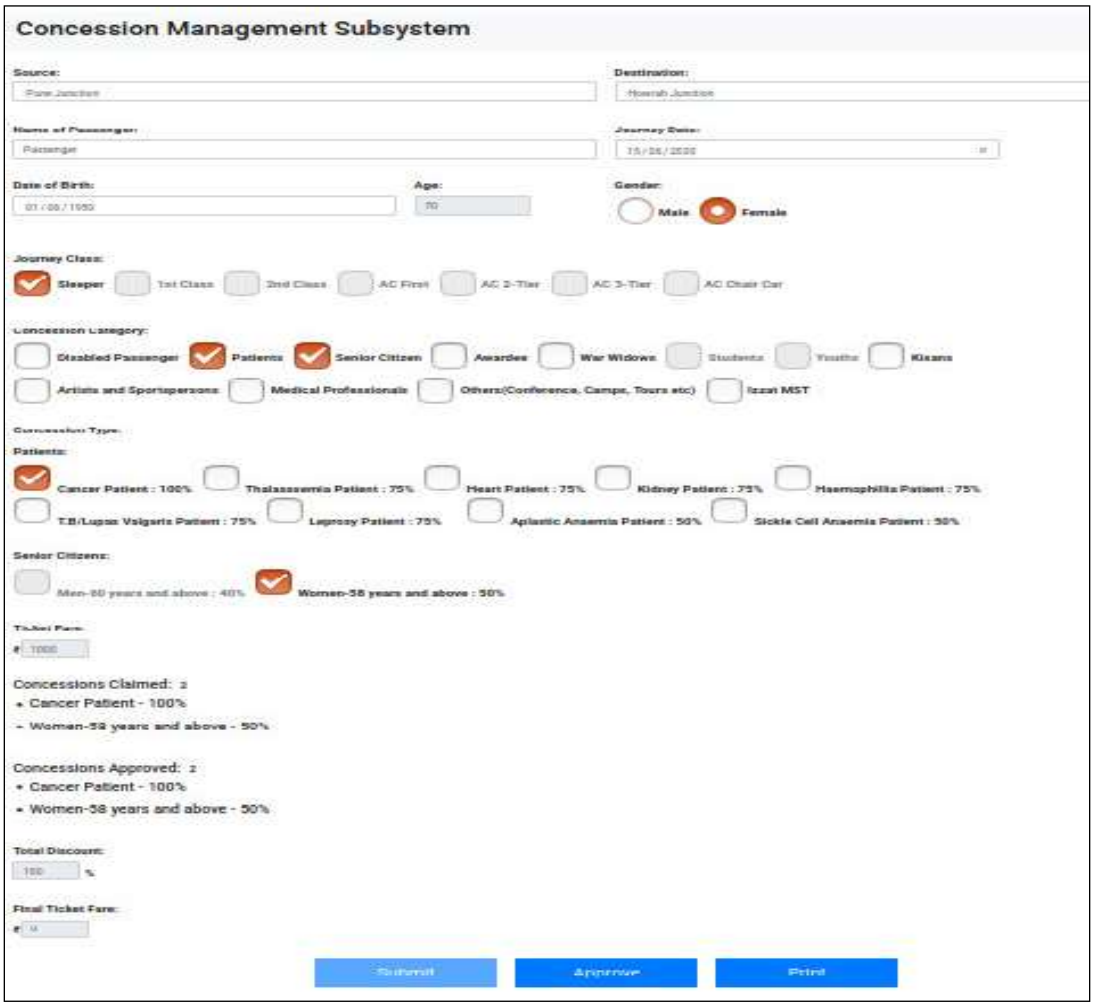

Fig. 1. GUI of CMSS.

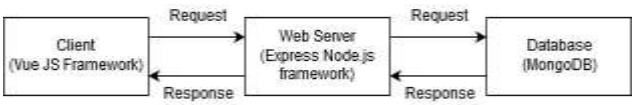

Fig. 2. Client-Server Architecture of the Proposed CMSS.

A request is sent by the user at the client-side and a request object is generated. It represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers. Response to the request is generated at the server-side and it creates a response object. It represents the HTTP response that an Express app sends when it gets an HTTP request. MongoDB open-source document-based database is used in the proposed system to store the user and concession details. The system is implemented as per the specifications mentioned in Table I, CLO-ATDD rules for availing concessions mentioned in Table II and user acceptance tests mentioned in Table III.

2) Test case generation technique: Test case generation technique of the proposed system is explained in this section. The percentage of concession is dependent on journey class and concession categories which are mentioned in Table I. Only widow concession is gender-specific and so it applies to the females only. The fare of the ticket is based on Child and Adult passenger types. Children above the age of 5 years and under 12 years of age are charged as 50% of adult ticket fare and considered as a child passenger. The fare of the ticket is exempted for the children under five years of age. Necessary information details for reservation class are shown in Table IV. The concession categories, types, selection mode of concession types and feasibility details of proposed CMSS are shown in Table V. The concession categories are infeasible to passenger types, gender and age. Some concessions are infeasible to child passenger type and some are infeasible to an adult passenger type. The passengers can select multiple concession types in the disabled passenger and patient concession category. For the remaining concession types, only one concession type is selected.

3) Proposed algorithm to generate the test cases: Step 1:- A dictionary in python for all types of journey class declared named class_dict and a dictionary for the type of

passenger names type_dict.

e.g. type_dict={"1":"Child","2":"Adult"}

TABLE IV. BASIC INFORMATION DETAILS FOR RESERVATION CLASS

Journey Class	SL	2ND	1ST	1AC	2AC	3AC	AC Chair
Gender	Male	Female					
Passenger Type	Child (5<=Age<= 12 yrs)	Adult (Age> 12 yrs)					

TABLE V. CONCESSION CATEGORIES AND TYPES ALONG WITH SELECTION MODE BASED ON PASSENGER TYPES

Concession categories	Disabled Passenger	Patient	Senior Citizen	Widow	Student	Awardee	Artists & Sports-person	Youth	Kisan	Medical Professionals
No. of concession types	04	10	2	5	8	5	5	4	3	2
Selection Mode	Multiple	Multiple	Single	Single	Single	Single	Single	Single	Single	Single
Applicable to passenger type	Adult, Child	Adult, Child	Adult (based on age)	Adult female	Adult	Adult	Adult	Adult	Adult	Adult

Step 2:- A dictionary of all the concession categories declared with keys against the values of the patient, disable passenger etc. A dictionary named categories is defined with the keys being broad categories like Patient, Disability etc. and the keys a dictionary which contains the specifications of that category, e.g. categories = {'type':type_dict}.

Step 3:- An empty dictionary concession_class declared where the keys of each concession type against a list of values of concession in order of journey class. e.g. concession_class["Cancer"] = [100, 75, 75, 50, 50, 100, 75].

Step 4:- A function concession defined to calculate the concession in case of multiple values for patients and disabled passenger category. This concession function takes in the list of disability, patients etc. as pairs for 2-way testing and as triplets for 3-way testing along with the journey class as resvClass. It then calculates the length of this list and sorts it in ascending order. This sorted list has concession values, the last element being the highest concession. Depending on the number of concession values, they are multiplied with a multiplier to calculate final concession and a final concession is returned through this function.

Step 5:- A function printCombination ( ) defined which takes in an array, n and r to calculate all combinations of size r in an array of size n (nCr).

Step 6:- A function combinationUtil ( ) defined which takes in the array, a temporary array named data, start which is the starting index of the array, end which is the end index of the array and r which is the size of combination. Both these functions are in the main function called getCombination ( ) which gives us a result of all combinations. All of these combinations are stored in the form of a dictionary where the keys are the broad categories i.e. Disability, Patient, Awardee etc. and the values are a list of all possible combinations of that category. e.g. {'Disability': [['Orthopedically Handicapped', 'Mentally retarded']}.

Step 7:- A final concession_list dictionary is declared whose keys are as same as the final result, i.e. the broad category. Furthermore, the values will be all the information for that broad category, i.e., the journey class, the type of customer, the combination of that broad category and the concession given.

Step 8:- Iterating through a loop, in the dictionary final_result, we go through the dictionary values to pick up each possible combination that we have calculated through getCombination ( ) function.

Step 9:- Another for loop iterates class_dict to calculate journey class to display for a particular combination and to go through type_dict to find out the type of customer for that combination.

Step 10:- Two lists are declared to append the list of all the combinations against the journey class and its respective concession.

Step 11:- The first list goes through the final_result, makes a list of journey class, type of customer, concession value and the particular combination. The second loop is used to take this entire list having information about a particular combination to store it in a different list called final which has values as list, i.e., temp_final.

Examples-

{'Disability': [['SL', 'Adult/Child', 78.75, ['Orthopedically Handicapped', 'Mentally retarded']], ['2nd', 'Adult/Child', 78.75, ['Orthopedically Handicapped', 'Mentally retarded']]}

Step 12:- This final list for a broad category is created and then stored in concession_list against its key i.e., Disability, Patient etc.

Step 13:- Edit concession_list according to particular conditions by using a loop, iterating through the final concession_list to make changes wherever it is needed. All the test cases having the same concession for all seven journey classes have been combined into one test case.

Step 14:- Writing all of the results into a text file by taking the console output and creating a new text file which contains all the test cases. The unsuccessful cases for each broad category are written at the end of the file.

V. RESULT ANALYSIS AND DISCUSSION

The authors proposed the technique of generation of combinatorial test cases to reduce the time and improve the effectiveness of the testing. Total of 665 test cases are generated using 2-way combinatorial strategy, and 1435 test cases are generated through 3-way combinatorial strategy. The authors claimed that proposed testing technique gives

reliability and efficiency completely. The user acceptance test is fully satisfied. The combinatorial explosion of test cases is avoided using CLO-ATDD model. The result of the proposed technique is shown in Table VI.

A. Findings

There are some systems such as Reservation system, College admission system, Concession management system, etc. where combinatorial logic plays an extremely important role. The authors made a survey to find a suitable model among the classical process models viz. Waterfall, Spiral, Incremental, etc., and agile process models viz. Extreme Programming, Scrum, etc., to represent combinatorial logic. The authors found that TDD and ATDD models are more suitable to represent combinatorial logic. Compared to TDD model, ATDD is most suitable model. The authors proposed to enhance the existing ATDD model to CLO-ATDD model.

B. Comparision with Existing Models

In the TDD model, test cases are written in the same language in which the features are implemented. The TDD model focuses on the implementation of the features. In the ATDD model, test cases are written in simple business language. User acceptance tests are written from the user's point of view. Developers implement the system using these user acceptance tests. In CLO-ATDD model, user acceptance tests are prepared in a business language using combinatorial logic oriented rules. Hence, CLO-ATDD is more suitable model than TDD and ATDD to incorporate combinatorial logic.

C. Limitations of the Proposed Model

The first limitation is generating more than 6-way combinations of parameters for combinatorial testing. In case, if we have to go for higher values of interaction strength, we may have to find new algorithms and techniques. The second limitation is exhaustive testing. But, if the size of test suite is small and manageable, then we recommend exhaustive testing. The third limitation is combinatorial explosion problem as discussed in section (III (B(c))) which is a universal problem in software testing. Our proposed model attempts to avoid this combinatorial explosion problem to some extent.

TABLE VI. RESULTS OF PROPOSED TECHNIQUE

No. test cases/test case generation	**Combinatorial test case generation**	
	2-way	3-way
Number of test cases generated	665	1435

VI. CONCLUSION AND FUTURE WORK

The present paper has explored an enhancement of the Acceptance Test Driven Development model to improve the software development life cycle by applying combinatorial logic. The proposed Combinatorial Logic Oriented-Acceptance Test Driven Development (CLO-ATDD) model has been illustrated with a case study of the Indian Railway Reservation System. This system was fully implemented and verified by using the combinatorial testing and combinatorial testing approach. This system typically considers multiple concession types and calculates an appropriate percentage of concession in ticket fare to the passenger as per proposed CLO-ATDD model. The authors hope that this proposal would be valuable for end-user, i.e. the tourism community ultimately. In future, the aforementioned CLO-ATDD model can be applied to design various systems where combinatorial logic is essential. Also, this model will also helpful to generate combinatorial logic oriented test cases using UML diagrams.

REFERENCES

[1] Lei, Yu, and Kuo-Chung Tai. "In-parameter-order: A test generation strategy for pairwise testing." In Proceedings Third IEEE International High-Assurance Systems Engineering Symposium (Cat. No. 98EX231), pp. 254-261 (1998).

[2] Grindal, Mats, Jeff Offutt, and Sten F. Andler. "Combination testing strategies: a survey." Software Testing, Verification and Reliability 15, no. 3 (2005): 167-199.

[3] Nie, Changhai, and Hareton Leung. "A survey of combinatorial testing." ACM Computing Surveys (CSUR) 43, no. 2 (2011): 1-29.

[4] Kuhn, D. Richard, Renee Bryce, Feng Duan, Laleh Sh Ghandehari, Yu Lei, and Raghu N. Kacker. "Combinatorial testing: Theory and practice." In Advances in Computers, vol. 99, pp. 1-66. Elsevier, 2015.

[5] Lott, C., Ashish Jain, and S. Dalal. "Modeling requirements for combinatorial software testing." In ACM SIGSOFT Software Engineering Notes, vol. 30, no. 4 (2005), pp. 1-7.

[6] Zamansky, Anna, and Eitan Farchi. "Helping the tester get it right: Towards supporting agile combinatorial test design." In SEFM 2015 Collocated Workshops, pp. 35-42. Springer, Berlin, Heidelberg, 2015.

[7] Lei, Yu, and Kuo-Chung Tai. "In-parameter-order: A test generation strategy for pairwise testing." In Proceedings Third IEEE International High-Assurance Systems Engineering.

[8] Mondal, Sayani, and Partha Pratim Das. "Effectiveness of Test-Driven Development as an SDLC Model: A case study of an elevator controller design." In Emerging Trends in Computing and Communication, pp. 225-233. Springer, New Delhi, 2014.

[9] Dogša, Tomaž, and David Batič. "The effectiveness of test-driven development: an industrial case study." Software Quality Journal 19, no. 4 (2011): 643-661.

[10] Cohen, David M., Siddhartha R. Dalal, Michael L. Fredman, and Gardner C. Patton. "The AETG system: An approach to testing based on combinatorial design." IEEE Transactions on Software Engineering Vol.23, No. 7 (1997), pp. 437-444.

[11] R. Kuhn, Yu Lei and Raghu Kacker, “Practical Combinatorial Testing: beyond Pair wise”, IEEE Computer Society - IT Professional, Vol. 10, No. 3 (2008).

[12] D. Richard Kuhn, Raghu N. Kacker and Yu Lei, “Practical combinatorial testing”, NIST Special Publication, (2010).

[13] Bhuvana, S., and M. V. Srinath. "A survey on Automated Combinatorial Testing for Software Tool (ACTS) with experimental revise based on T-way test generation." (2016).

[14] Jayaram, Rekha, and R. Krishnan. "Approaches to Fault Localization in Combinatorial Testing: A Survey." In Smart Computing and Informatics, pp. 533-540. Springer, Singapore, 2018.

[15] Khalsa, Sunint Kaur, and Yvan Labiche. "An orchestrated survey of available algorithms and tools for combinatorial testing." In 2014 IEEE 25th International Symposium on Software Reliability Engineering, pp. 323-334.

[16] Mudarakola, Lakshmi Prasad, and M. Padmaja. "The survey on artificial life techniques for generating the test cases for combinatorial testing." International Journal of Research Studies in Computer Science and Engineering (IJRSCSE) 2, no. 6 (2015): 19-26.

[17] Mudarakola, Lakshmi Prasad, J. K. R. Sastry, and V. Chandra Prakash. "Testing embedded systems using test cases generated through combinatorial techniques." International Journal of Engineering & Technology 7, no. 2.7 (2018): 146-158.

[18] Vudatha, Chandra Prakash, Sateesh Nalliboena, Sastry Kr Jammalamadaka, Bala Krishna Kamesh Duvvuri, and L. S. S. Reddy.

"Automated generation of test cases from output domain of an embedded system using Genetic algorithms." 3rd International In Electronics Computer Technology (ICECT), IEEE (2011), vol. 5, pp. 216-220.

[19] Vudatha, Chandra Prakash, Sateesh Nalliboena, Sastry KR Jammalamadaka, Bala Krishna Kamesh Duvvuri, and L. S. S. Reddy. "Automated generation of test cases from output domain and critical regions of embedded systems using genetic algorithms." *2nd National Conference on Emerging Trends and Applications in Computer Science*, pp. 1-6. IEEE, 2011.

[20] Vudatha Chandra Prakash, Sastry K R Jammalamadaka, and Bala Krishna Kamesh Duvvuri. "Automated generation of Test cases for testing critical regions of embedded systems through Adjacent Pair-wise Testing." International Journal of Mathematics and Computational Methods in Science & Technology Vol.2, No.2, (2012), pp. 10-15.

[21] Wu, Huayao, Changhai Nie, Justyna Petke, Yue Jia, and Mark Harman. "A Survey of Constrained Combinatorial Testing." (2019).

[22] Ramgouda Patil, V Chandra Prakash, "Neural Network Based Approach for Improving Combinatorial Coverage in Combinatorial Testing Approach", Journal of Theoretical and Applied Information Technology, Vol.96. No 20 (2018),pp.6677-6687

[23] Gouda, Ram, and V. Chandraprakash. "Optimization Driven Constraints Handling in Combinatorial Interaction Testing." International Journal of Open Source Software and Processes (IJOSSP) 10, no. 3 (2019): 19-37.

[24] Ramgouda, P., and V. Chandraprakash. "Constraints handling in combinatorial interaction testing using multi-objective crow search and fruitfly optimization." Soft Computing 23, no. 8 (2019): 2713-2726.

[25] Lott, C., Ashish Jain, and S. Dalal. "Modeling requirements for combinatorial software testing." In ACM SIGSOFT Software Engineering Notes, vol. 30, no. 4 (2005), pp. 1-7.

[26] Grindal, Mats, and Jeff Offutt. "Input parameter modeling for combination strategies." In Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering, pp. 255-260. ACTA Press, 2007.

[27] V.Chandra Prakash and Kadiyala Priyanka, 2016. "Test Case Generation for Pairwise + Testing." Asian Journal of Information Technology. Vol. 15 No.23 (2016), pp.4800-4805.

[28] Vrushali Kondhalkar, V.Chandra Prakash. "Automated Generation of Test Cases for Conducting Pairwise plus Testing". Journal of Advanced Research in Dynamical and Control Systems (ISSN 1943-023X), Issue 7 (2018), pp.1484-1492.

[29] Dr.V.Chandra Prakash, Subhash Tatale, Vrushali Kondhalkar, Laxmi Bewoor. "A critical review on automated test case generation for conducting combinatorial testing using particle swarm optimization." International Journal of Engineering & Technology (UAE), Vol.7, No.3.8, (2018), pp. 22-28.

[30] Dhadyalla, Gunwant, Neelu Kumari, and Timothy Snell. "Combinatorial testing for an automotive hybrid electric vehicle control system: a case study." In Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on, pp. 51-57.

[31] Dr.ChandraPrakash V, Dr.Sastry JKR, Sravani G, Manasa USL, Khyathi A, Harini A, Testing Software Through Genetic Algorithms – A Survey, Journal of Advanced Research in Dynamical and Control Systems Vol. 9. Sp– 12. (2017).

[32] M. Lakshmi Prasad,Dr.J.K.R. Sastry, A Graph Based Strategy (GBS) for Generating Test Cases Meant for Testing Embedded Systems Using Combinatorial Approaches, Jour of Adv Research in Dynamical & Control Systems, Vol. 10, 01-Special Issue, (2018).

[33] J. Sasi Bhanu, M. Lakshmi Prasad, Dr. JKR Sastry, A Combinatorial Particle Swarm Optimization (PSO) Technique for Testing an Embedded System, Jour of Adv Research in Dynamical & Control Systems, Vol. 10, 07-Special Issue, 2018, pp. (321-336).

[34] M. Lakshmi Prasad, Dr. JKR Sastry, Building Test Cases by Particle Swarm Optimization (PSO) For Multi Output Domain Embedded Systems Using Combinatorial Techniques, Jour of Adv Research in Dynamical & Control Systems, Vol. 10, 06-Special Issue, 2018.

[35] Dr. J Sasi Bhanu,M. Lakshmi Prasad, Dr. J. K. R. Sastry, Combinatorial Neural Network Based a Testing of an Embedded System, Jour of Adv Research in Dynamical & Control Systems, Vol. 10, 07-Special Issue, 2018.

[36] Dr. J Sasi Bhanu, M. Lakshmi Prasad, Dr. J. K. R. Sastry, Testing Embedded System through Optimal Combinatorial Mining Technique, Jour of Adv Research in Dynamical & Control Systems, Vol. 10, 07-Special Issue, 2018, pp. (337-354).

[37] Dr. J Sasi Bhanu, M. Lakshmi Prasad, Dr. J. K. R. Sastry, Testing Embedded Systems Using - A Graph Based Combinatorial Method (GBCM), Jour of Adv Research in Dynamical & Control Systems, Vol. 10, 07-Special Issue, 2018, pp. (355-375).

[38] M. Lakshmi Prasad, A. Raja Sekhar Reddy, J.K.R. Sastry, GAPSO: Optimal Test Set Generator for Pairwise Testing, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8 Issue-6, August 2019.

[39] Dr.Sasi BhanuJ, Dr.Baswaraj D, Sunitha Devi Bigul, Dr. JKR Sastry, Generating Test cases for Testing Embedded Systems using Combinatorial Techniques and Neural Networks based Learning Model, International Journal of Emerging Trends in Engineering Research, Volume 7, No. 11 November 2019, pp 417-429.

[40] Moe, Myint Myint. "Comparative Study of Test-Driven Development (TDD), Behavior-Driven Development (BDD) and Acceptance Test–Driven Development (ATDD)." (2019).

[41] Losada, Begoña, Juan-Miguel López-Gil, and Maite Urretavizcaya. "Improving Agile Software Development Methods by means of User Objectives: An End User Guided Acceptance Test-Driven Development Proposal." In Proceedings of the International Conference on Human Computer Interaction, pp. 1-4. 2019.

[42] Hoffmann, Luiz Felipe Simoes, Luiz Eduardo Guarino de Vasconcelos, Etiene Lamas, Adilson Marques da Cunha, and Luiz Alberto Vieira Dias. "Applying acceptance test driven development to a problem based learning academic real-time system." In 2014 11th International Conference on Information Technology: New Generations, pp. 3-8. IEEE.

[43] Basit, Mujeeb A., Krystal L. Baldwin, Vaishnavi Kannan, Emily L. Flahaven, Cassandra J. Parks, Jason M. Ott, and Duwayne L. Willett. "Agile Acceptance Test–Driven Development of Clinical Decision Support Advisories: Feasibility of Using Open Source Software." JMIR medical informatics 6, no. 2 (2018).

[44] Clerissi, Diego, Maurizio Leotta, Gianna Reggio, and Filippo Ricca. "A lightweight semi-automated acceptance test-driven development approach for web applications." In International Conference on Web Engineering, pp. 593-597. Springer, Cham, 2016.

[45] Nilsson, Johan, and Xiaoqian Xiong. "Applicability of Acceptance Test Driven Development in Integration and Verification Process in a Large Scale Company." (2016).

[46] Ramler, Rudolf, and Claus Klammer. "Enhancing Acceptance Test-Driven Development with Model-based Test Generation." In 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 503-504.

[47] Anang, Yunarso, Masakazu Takahashi, and Yoshimichi Watanabe. "Collaborating Acceptance Test-Driven Development and QFD in Business System Software Development." (2017).

[48] Connolly, David, Frank Keenan, and Fergal McCaffery. "Acceptance test-driven development by annotation of existing documentation." (2010).

[49] Aggarwal, Vishal, and Manpreet Singh. "Acceptance Test Driven Development." Journal of Advanced Computing and Communication Technologies (ISSN: 2347-2804) (2014): 1-4.

[50] Weiss, Johannes, Peter Mandl, and Alexander Schill. "Introducing the QCEP-testing system for executable acceptance test driven development of complex event processing applications." In Proceedings of the 2013 International Workshop on Joining AcadeMiA and Industry Contributions to testing Automation, pp. 13-18.

[51] Axelrod, Arnon. "Acceptance Test Driven Development." In Complete Guide to Test Automation, pp. 371-394. Apress, Berkeley, CA, 2018