International Journal of Information Technology and Management
Vol. 12, Issue No. 1, February-2017, ISSN 2249-4510

The Test Cases Generation: Approaches and Techniques in Software Testing

Khadija Sania Ahmad[1]* Shaista Khan[2] Priyanka[3] Nazia Ahmad[4]

[1]M. Tech in CSE Scholar, Al-Falah University, Dhauj, Faridabad, Haryana

[2]M. Tech in CSE Scholar, Al-Falah University, Dhauj, Faridabad, Haryana

[3]M. Tech in CSE Scholar, Al-Falah University, Dhauj, Faridabad, Haryana

[4]IMAM Abdulrahman Bin Faisal University (Formerly University of DAMMAM)

Abstract – The design of an appropriate test suite for software testing is a challenging task. It requires a suitable tradeoff between effectiveness, e.g., a sufficient amount of test cases to satisfy the test goals of a given coverage criterion, and efficiency, e.g., a redundancy-reduced selection of test cases. Recent test suite optimization approaches, therefore, usually require an explicit enumeration of existing test cases. The test suite design for covering entire software product lines was even more problematic as the dependency between test cases, test goals and product configurations has to be taken into account. Due to the exponential number of configurations with respect to the number of features, an explicit enumeration of all products for optimizing a product-line test suite is impartibly.

Keywords: Test Case, software testing, model, UML

----------X----------

1. INTRODUCTION

Testing is the procedure of evaluating a system or its part(s) with the intention to find whether it satisfies the specified requirements or not. In simple words, testing is performing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

Software testing is executed to confirm that the completed software package functions according to the expectations defined by the necessities/specifications. The overall objective of software testing is not only to find out every software bugs that lives but also to expose situations that could negatively impact the customer, usability and/or maintainability.

From the module level to the application level, this article describes the different types of testing. Depending upon the reason for testing and the software requirements/specs, a combination of testing methodologies is applied.

Test automation is really two different things testing, which is one discipline, and automation, which is another. Automating software testing is no different than automate secretarial or any other business function: in each case, a computer is being instructed to perform a task before performed manually. Whether these commands are stored in something called a script or a program, they both have all of the characteristics of source code.

Requirements can be appointed needs and used to quantify availability for discharge. Having Requirements attached to tests likewise diminishes bewilderment about which Requirements have been fulfilled or fizzled in light of the consequences of the test, in this manner make less difficult the test and mistake log reports. A Requirements network is a method for recognition track of which prerequisites have an associated test. A commitment that has excessively numerous tests might be too for the most part characterized, and ought to be separated into discrete occasions, or it might essentially have a greater number of tests than are expected to take care of business. Then again, a test that is related with an excessive number of prerequisites might be excessively mind boggling and ought to be separated into littler, detach tests that are more focused to exact Requirements.

There are two fundamental methodologies: initial one is tending to every conceivable blend, and one that depends on tending to the littlest sum conceivable mixes. Utilizing the previous technique, Requirements are less demanding to characterize since

interdependencies are not as unsafe, but rather the quantity of tests delivered is more noteworthy. The last strategy produces less tests, yet needs a more muddled methods for characterizing needs so that connections among them are attested with the arithmetical exactitude expected to enhance the quantity of tests.

Relapse testing is performed when changes are made to existing programming; the motivation behind relapse testing is to give certainty that the recently presented changes don't block the practices of the current, unaltered piece of the product. It is a mind boggling methodology that is all the all the more difficult as a result of a portion of the current patterns in programming improvement ideal models. For instance, the segment based programming improvement strategy tends to bring about utilization of many discovery segments, frequently embraced from an outsider. Any adjustment in the outsider segments may meddle with whatever is left of the product framework, yet it is difficult to perform relapse testing on the grounds that the internals of the outsider segments are not known to their clients. The shorter life cycle of programming advancement, for example, the one proposed by the nimble programming discipline additionally forces confinements and limitations on how relapse testing can be performed inside constrained assets. Actually, the clearest way to deal with this issue is to just execute all the current experiments in the test suite; this is known as a retest-all approach. Be that as it may, as programming advances, the test suite has a tendency to develop, which implies it might be restrictively costly to execute the whole test suite. These restriction strengths thought of systems that look to lessen the exertion required for relapse testing in different ways. Various diverse methodologies have been contemplated to help the relapse testing process. The three noteworthy branches incorporate test suite minimization, experiment determination and experiment prioritization. Test suite minimization is a procedure that tries to recognize and after that dispense with the old or excess experiments from the test suite. Experiment determination manages the issue of choosing a subset of experiments that will be utilized to test the changed parts of the product. At last, experiment prioritization concerns the distinguishing proof of the "perfect" requesting of experiments that augments attractive properties, for example, early blame discovery. Existing experimental reviews demonstrate that the utilization of these methods can be cost-effective.

2. REVIEW OF LITERATURE

Another extraordinary approach to keep up the adequacy of the relapse test suite is to have a decent following system between the components being worked on. This ought to be a reliable movement keeping in mind the end goal to keep up the test suites successfully as it would help test administrator to confirm the component agenda and approve the test scope for an element that is being created in the discharge.

The adequacy of the relapse test suite can be effortlessly kept up by checking the progressions to the test suite. An obviously plot process will guarantee that exclusive tests that are valuable to the whole testing methodology get added to the test suite, which guarantees the effectiveness and ease of use of the test tackle at an abnormal state.

Considering occasional cleanup of old tests is another awesome way to deal with keep up adequacy of an element rich relapse test suite. In this situation, all the current tests in the test suite should be broke down for their viability in a particular situation. Additionally, there will be situations where certain components won't be bolstered in light of the diverse item bearing. In such cases, the pertinent relapse test suites ought to likewise be backed off. It will guarantee power of the relapse test suite for a drawn out stretch of time.

We can likewise gauge the viability of relapse test suites on a discharge to-discharge premise. It will permit us to know the main driver for diminishment in the viability of the test outfit assuming any, and empower us to make fitting move on the same.

Gathering of a few measurements and their investigation could likewise be helpful with regards to the viability of the relapse test suite. It will help we get great perceivability on the viability of the relapse test suite. We can consider distinctive measurements, for example, rate of imperfections found by the relapse tests suite, their significance, and so on.

In spite of the immense interest in testing said above, late data from Capers Jones demonstrates that the distinctive sorts of testing are generally insufficient. Specifically, testing commonly just distinguishes from one-fourth to one-portion of deformities, while other confirmation techniques, for example, examinations, are regularly more successful s. Deficient testing is one of the fundamental reasons why programming is commonly conveyed with around 2 to 7 deserts for every thousand lines of code (KLOC). While this may appear like an irrelevant sum, the outcome is that real programming dependent frameworks are being conveyed and put into operation with hundreds or even a huge number of lingering imperfections. In the event that product vulnerabilities, (for example, the CWE/SAN Top 25 Most Dangerous Software Errors) are considered security abandons, the rates are much all the more upsetting.

Plainly, there are real issues with the productivity and adequacy of testing as it is at present performed practically speaking.

The substantial number of testing issues required that they be ordered. At the top level, these issues were sorted out into two gatherings

General testing issues that are not particular to a testing, but rather apply to every diverse sort of testing.

Test sort particular issues that are particular to a solitary kind of testing, for example, unit testing, combination testing, and framework testing.

Test arranging and booking issues frequently happen when there is no different test arrange, yet rather exceedingly deficient and shallow outlines in other arranging records. Test arrangements are regularly overlooked once they are composed, and experiment portrayals are frequently confused for general test arranges. The calendar of testing is frequently deficient for the measure of testing that ought to be performed, particularly when testing is fundamentally manual. Huge testing is frequently put off until past the point of no return in the advancement procedure, particularly on tasks utilizing customary successive improvement cycles.

Partner inclusion and responsibility problems include having the wrong testing attitude (that the reason for testing is to demonstrate that the product works as opposed to discovering absconds), having implausible testing desires (that testing will discover the majority of the huge imperfections), and having partners who are deficient dedicated to and supporting of the testing exertion.

3. SOFTWARE TEST CASE GENERATION TECHNIQUUES

The aggregate cost, time and exertion required for general testing relies on upon aggregate number of experiments. An experiment is an arrangement of sources of info given to the product or application to get the pre-indicated yield. The exertion fundamentally relies on upon the measure of the application and the quantity of test cases.R.P. Mohapatra and Jitendra Singh depict the well ordered strategy for experiment era system.

The initial step is to locate every conceivable limitation all the way hubs. A requirement is a couple of mathematical expressions, which manage states of factors amongst begin and complete hubs.

To diminish the experiments, the most elevated esteem is allocated to the variable having greatest esteem and the least esteem is relegated to least an incentive inside its predetermined range.

After this, the steady esteem is alloted to the given variable at every hub in the characterized way.At long last table is made that incorporates all conceivable experiments. Leung and White sort test cases into five classes as reusable, retest capable, out of date, basic, new detail and new auxiliary experiments. Reusable experiments just execute the parts of the program that stay unaltered between two forms. Re-testable experiments execute the parts of a program that have been changed in another program. Out of date Test Cases can be rendered out of date on the grounds that their information/yield connection is do not right anymore, because of changes in details and they don't test what they were intended to test because of alterations in the program. There are a few techniques for experiment era that relies on upon the application like experiment era for web application, question situated application, organized based frameworks, UML applications, applications in view of transformative and hereditary calculations and numerous others. Utilizing some arrangement of data sources tests programming and its prosperity relies on upon the normal yields got from the experiment conditions. Consistently, a few distinctive testing have been proposed for creating experiments.

Test suite minimization methods plan to recognize repetitive experiments and to expel them from the test suite with a specific end goal to lessen the measure of the test suite. The minimization issue depicted by Definition 1 can be considered as the negligible hitting set issue. Take note of that the negligible hitting set detailing of the test suite minimization issue relies on upon the suspicion that every ri can be fulfilled by a solitary experiment. By and by, this may not be valid. For instance, assume that the test prerequisite is utilitarian instead of basic and, in this manner, requires more than one experiment to be fulfilled. The insignificant hitting set plan does not make a difference anymore. With a specific end goal to apply the given plan of the issue, the useful granularity of experiments should be balanced as needs be. The conformity procedure might be either that a larger amount of reflection would be required so that each experiment prerequisite can be met with a solitary test situation made out of applicable experiments, or that an "extensive" useful necessity should be isolated into littler sub-prerequisites that will compare to individual experiments.

The NP-fulfillment of the test suite minimization issue supports the use of heuristics; past work on experiment minimization can be viewed as the improvement of various heuristics for the negligible hitting set issue. Jeffrey and Gupta broadened the HGS heuristic so that specific experiments are specifically held. This 'particular excess' is acquired by presenting an optional arrangement of testing

prerequisites. At the point when an experiment is set apart as repetitive regarding the principal set of testing necessities, Jeffrey and Gupta considered whether the experiment is additionally excess as for the second arrangement of testing prerequisites. On the off chance that it is not, the experiment is as yet chosen, bringing about a specific level of excess as for the main arrangement of testing prerequisites. The experimental assessment utilized branch scope as the principal set of testing prerequisites and all-utilizes scope data acquired by information stream examination. The outcomes were contrasted with two renditions of the HGS heuristic, in light of branch scope and def-utilize scope. The outcomes demonstrated that, while their procedure created bigger test suites, the blame recognition capacity was better saved contrasted with single-rule renditions of the HGS heuristic. Though the specific excess approach considers the auxiliary foundation just when an experiment is set apart as being repetitive by the principal model, Black et al. considered a bi-criteria approach that considers both testing criteria.

4. APPROACHES IN TESTING GENERATION

Once an application has been characterized, it should be exhibited to clients. On account of our sun oriented power sweater, one clear and generally straightforward method for showing it to clients is to depict it as far as appearance, usefulness, and advantages. It won't be that straightforward for clients, in any case, to really envision the advantages of the sweater on the premise of a portrayal as it were. For instance, by what means will they know whether the sweater is agreeable and not very warm? A moment, considerably more refined method for showing the sweater to clients is to have them attempt on a model. Clearly, this requires the sweater model have as of now been produced. This is commonly not the situation, however. In the early advancement period of an achievement innovation advancement, the new innovation either does not exist yet, in light of the fact that it is not (yet) completely operational, or it would be too expensive to build up a model. All things considered, organizations require the early information with respect to the conceivable estimation of an application and in regards to the bearing into which an application ought to be produced. So as to have the capacity to give substantial info, Klink and Athaide contend that clients ought to get however much boost material as could reasonably be expected that mirrors the sort of data that would be accessible in the commercial center. The majority of this implies an introduction strategy is required in which utilizations of innovative leaps forward can be exhibited to clients before they have been produced in detail.

A critical normal for such an introduction technique ought to be that it fortifies clients' creative ability. That is, the technique ought to prompt clients to envision the future estimation of the cutting edge development. Early idea accounts, a particular kind of future item situations, are especially encouraging in this regard. In particular, an early idea account is a portrayal and additionally delineation in which somebody utilizes another item (idea) in a particular setting. It incorporates the utilization circumstance of the item, its advantages, and its traits. It regularly appears as a story in which a client uses the new item in a future setting, taking after an exemplary storyline with a presentation and a completion. The story can likewise be joined by visual material that shows different plan parts of the item and its expected condition. In doing as such, stories bring out symbolism clients can envision an innovation and its potential advantages by observing it connected in an item, which utilize is made express. In our illustration, such a story would portray/delineate a day from the life of the primary character of the account (for example, Helen), who utilizes the sun based power sweater to charge her mobile phone while cycling to work, to charge her PDA amid a meeting, and to power her MP3 player while running through the recreation center after work. The quality of the early idea account technique is that it helps clients to look ahead and envision how the new item could be gainful to them, notwithstanding when there is no genuine item or model accessible, when clients are deficient with regards to learning about the development, and when they are encountering extraordinary vulnerability about the leap forward innovation and its applications. A few organizations have comprehended the capability of early idea stories and are utilizing them to clarify their new items. Vodafone (www.vodafone.com), for example, has a future area on its site, recounting a story and vivifying a few potential outcomes of their future innovation. This empowers their clients to look ahead, to imagine solid settings in which the new innovation can be utilized, and to consider potential outcomes of the new innovation for everyday life. Early idea accounts are proposed to be viable in light of the fact that they fortify clients to envision the circumstance depicted in the story, which empowers them to give substantial item judgments. Without a doubt, past research bolsters the claim that the utilization of visual symbolism in statistical surveying causes clients' reactions to be more legitimate even with vulnerability. A few reviews have demonstrated that the evoked visual symbolism can bolster a client's item assessment and make it less negative. For instance, Hoeffler demonstrated that when clients need involvement and learning about another item's traits and advantages, they regularly utilize mental reenactment (i.e., creative energy) to make deductions about these dubious properties and advantages to foresee the advancement's utility.

5. SPECIFICATION BASED TEST CASE GENERATION TECHNIQUES

Particular based frameworks are techniques to create a game plan of investigations from detail records, for instance, a formal Requirements assurance. Undoubtedly, the detail accurately depicts what the structure is to oversee without

delineating how to do it. Subsequently, the item test manufacture has basic information about the item's handiness without separating it from silly inconspicuous components. The upsides of this system join that the Specification record can be used to decide expected results for test data and that tests may be made at the same time with diagram and execution. The latter is moreover profitable for breaking "Code now test later" practices in programming building and for making parallel testing practices for all stages (Subraya and Subrahmanya, 2000).

The Specification need record can be used as a purpose behind yield checking, on a very basic level diminishing one of the genuine costs of testing. Particulars can in like manner be destitute down with respect to their testability (Memon et al., 1999). The path toward delivering tests from the points of interest will habitually help the test fabricate discover issues with the judgments themselves. In case this movement is done early, the issues can be abstained from in front of timetable, saving time and resources. Creating tests in the midst of progression furthermore allows testing activities to be moved to a preceding some portion of the change methodology, thinking about more effective organizing and use of advantages. Test period can be self-sufficient of a particular use of the Specifications (Offutt et al., 1999).

Also, the detail based framework offers a less perplexing, sorted out and more formal approach to manage the headway of viable tests than non-Specification based testing strategies do. The strong relationship among Specification and tests finds faults and can enhance backslide testing. A basic usage of subtle elements in testing is to give test prophets.

The detriments of the assurance based framework with formal techniques are:

(1) The inconvenience of coordinating formal examination and the evident or genuine outcome in wander spending arrangement. Testing is a liberal piece of the item spending arrangement and formal techniques offer an opportunity to on a very basic level reduce testing costs, in this way making formal procedures additionally charming from the spending perspective (Liu et al., 2001) and (2) There is more unmistakable manual effort or methods in making tests, differentiated and frameworks including customized time shapes.

6. PORTRAY DIAGRAM-BASED TEST CASE GENERATION TECHNIQUES

Depict plot based methodologies are procedures to deliver test cases from model diagrams like UML Use Case chart. The going with entries concentrate current draw outline based investigation time frameworks that have been proposed for standard and electronic application for a long time.

An essential good position of model-based VandV is that it can be adequately automated, saving time and resources. Distinctive central focuses are moving the testing activities to an earlier piece of the item change handle and making tests that are free of a particular use of the arrangement (Javed et al., 2007). The going with segments portray existing assurance based techniques that have been proposed since 2000.

Heumann (2001) showed how using use cases to make test cases can help dispatch the testing strategy in front of plan for the change lifecycle and moreover help with testing framework. In an item change broaden, use cases describe structure programming necessities. Use case change begins immediately, so bona fide use cases for key thing handiness are available in early cycles. As showed by the Rational Unified Process (RUP), a use case is used to totally depict a gathering of exercises performed by a structure to give a discernible delayed consequence of noteworthy worth to a man or another system using the thing being taken a shot at. Use cases prompt the customer what's in store, the fashioner what to code, the specific creator what to chronicle and the analyzer what to test. He proposed three-organize method to make test cases from a totally ordered uses case:

(1) For every use case, create a full game plan of usage case circumstances

(2) For each circumstance, recognize no short of what one investigation and the conditions that will make it execute and

(3) For each analysis, perceive the data values with which to test.

Ryser and Glinz (2000) brought the helpful issues up in programming testing as takes after:

(1) Absence of masterminding/time and cost weight,

(2) Absence of test documentation,

(3) Absence of equipment support,

(4) Formal vernacular/Specification testing tongues required,

(5) Absence of measures, estimations and data to gauge testing and survey test quality and

(6) Lacking test quality.

Their proposed approach to manage settle the above issues is to get test cases from circumstances/UML use cases and state outlines. In their work, the time of tests is done in three stages:

(1) Preparatory examination and test game plan in the midst of circumstance creation,

(2) Experiment period from State outline and dependence charts and

(3) Test set refinement by application subordinate frameworks (regular, experience based testing).

Nilawar and Dascalu (2003) were enthused about testing on the web applications. Online applications are of creating multifaceted nature and it is a certified business to test them precisely. They focused on revelation testing which enables the item testing authorities to derive sets of data conditions that will totally rehearse each and every utilitarian essential. They assumed that revelation testing is all the more all things considered suitable and more imperative for web applications than various sorts of use. Plus, they proposed four phases to deliver test cases, in perspective of J. Heumann's four-phases (Heumann, 2001), as takes after:

(1) Organize use cases in perspective of the need traceability matrix,

(2) Create likely sufficient use cases and test circumstances,

(3) For each circumstance, recognize no short of what one investigation and the conditions and

(4) For each analysis, perceive test data values.

They furthermore showed that the investigation contains: a course of action of test information sources, execution conditions and expected results made for a particular objective.

Sinha and Smidts (2005) depicted another model based testing methodology made to perceive fundamental zone requirements. The new technique relies on upon showing the structure under test using a particularly Domain Specific Language (DSL). In the new framework, information about space Specification essentials of an application are discovered actually by manhandling properties of the DSL and are along these lines introduced in the test appear. The new system is associated with make test cases for the applications interfacing with social databases and the case DSL. Test suites delivered using the new techniques are upgraded with tests tending to space Specification comprehended necessities.

7. SOURCE CODE-BASED TEST CASE GENERATION TECHNIQUES

Source code-based techniques generally use control stream information to recognize a game plan of approaches to be secured and create appropriate trials for these ways. The control stream outline can be gotten from source code. The result is a plan of examinations with the going with association:

(1) Experiment ID,

(2) Test data,

(3) Test progression (generally called test steps),

(4) Expected result,

(5) Real result and

(6) Pass/crash and burn status.

The goings with segments delineates the source code-based frameworks that have been proposed since 1999.

8. RESEARCH CHALLENGES

Each investigation time procedure has feeble and strong concentrations, as tended to in the written work survey.

- Inefficient Test Case Generation Techniques with Limited Resources (e.g., Time, Effort and Cost): The item testing time of a wander is consistently allowed most diminished need. Ordinarily, programming testing engineers have a little measure of time, effort and cost to mastermind and arrangement explore, run test cases and survey test cases independently. Existing systems are not reasonable for complex applications with limited resources (e.g., time, effort and cost), both standard and web applications. An instance of a brain boggling web application is an application with component direct, heterogeneous portrayals, or novel control and data stream parts

- Lack of Ability to Identify Critical Domain Requirements: The present trial time techniques don't have the ability to address space specific necessities, because those essentials are not explicitly inspected in the assurance report. For an instance of this issue, where a procedure is proposed to make test cases for persistent systems

- Ignore Size of Test Case: Existing test time strategies intend to create test cases which extend cover for each circumstance. At times, they make generous trials which are

hard to execute given obliged time and resources

CONCLUSION

In this paper, the experiments era: methodologies and systems in software testing. We have primarily focused on test case generation of object-oriented software automatically. We have additionally investigated the strategy for utilization of developmental calculation like hereditary calculation to the programmed approach of testing. In this review we have proposed a way to deal with create test cases for question situated projects by utilizing UML action charts. We have utilized a heuristic manage to get the lessened experiments, which fulfill way scope as the test sufficiency criteria. In this paper, we have considered just the basic way for programmed experiment era. Our approach accomplishes the greatest branch scope and way scope, which is an additional preferred standpoint.

Our approach is not appropriate to deal with the huge and complex framework. This approach is particularly appropriate for straightforward frameworks where no more fork-joins, as settled fork joins and so on are included, which is our next goal. However our proposed framework is not adequate to deal with various sort of blunders, for example, work process mistakes, state based mistakes and so on. To conquer this bottleneck, a consolidated approach is basic and thus we have utilized the different UML outlines, for example, Activity, Class and Sequence chart. For our approach we have considered both Activity graph and Collaboration chart and we call them "air conditioning outline". We plan these two charts just for the situation of higher need. We utilize coordinated effort outline, in light of the fact that dissimilar to grouping chart, it demonstrates the connections among s and arrangement number of a message unequivocally. Coordinated effort outline is additionally equipped for taking care of more intricate fanning. The movement chart is utilized in light of its dynamic conduct of displaying. We can envision, build, indicate and archive the dynamic parts of a protest.

REFERENCES

Abraham Silberschatz, Henry F.Korth and S.Sudarshan, “Database system concepts”, Internation Edition ,2006,pp 739-741.

Ajitha Ranjan (2006). “Automated Requirements-Based test case Generation”. Communications of ACM.

Antonia Bertolino (2007). “Software testing Research: Achievements, challenges and dreams ”Future of Software Engineering.

B. N. Biswal, S. S. Barpanda and D. P. Mohapatra (2010). International Journal of Computer Applications, vol. 1, Issue 14.

David Alex Lamb (1988). “Software Engineering, planning for change,” Prentice Hall, Englewood Cliffs, NJ 07632, pp. 109– 112.

E. Hassan, A. Mockus, R. C. Holt, and P. M. Johnson (2005). Guest editor’s introduction: Special issue on mining software repositories. IEEE Trans. Softw. Eng., 31 (6): pp. 426–428.

Heumann, J. (2001). Generating test cases from use cases. Rational Software. http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jun01/GeneratingTestCasesFromUseCasesJune01.pdf.

J. G. Lee and C. G. Chung (2000). “An optimal representative set selection method”. Information and Software Technology, 42(1): pp. 17- 25.

Javed, A.Z., P.A. Strooper and G.N. Watson (2007). Automated generation of test cases using model-driven architecture. Proceeding of the Second International Workshop on Automation of Software Test, May 20 - 26, Minneapolis, USA, pp. 150-151.

K. Jain, M. N. Murty, and P. J. Flynn (1999). A Data clustering: review. ACM Computing Surveys, 31(3): pp. 264–323.

Lilly Ramesh (2009). “Knowledge Mining of Test Case System,” International Journal on Computer Science and Engineering, Vol.2(1), pp. 69-73.

M. J. Harrold, R. Gupta, and M. L. Soffa. (1993). A methodology for controlling the size of a test suit. ACM Trans. on Soft. Eng. and Meth., 2(3): pp. 270-285.

Mark Last and Menahem Friedman (2003). “The Data Mining approach to automated software testing.”.Communications of ACM.

Martina marre and Antonia Bertolino (2006). “using spanning sets for coverage testing”. IEEE transactions on software Engineering, vol.29.

Memon, A.M., M.E. Pollack and M.L. Soffa (1999). Using a goal-driven approach to generate test cases for GUIs. Proceedings of the 21st International Conference on Software

Engineering, May 16-22, Los Angeles, CA., USA., pp. 693-694.

Myra B. Cohen and Matthew B. Dwyer (2006). "Coverage and adequacy in software product line testing",Communications of ACM.

Offutt, A.J., Y. Xiong and S. Liu (1999). Criteria for generating specification-based tests. Proceedings of the 5th International Conference on Engineering of Complex Computer Systems, Oct. 18-22, Washington, USA., pp. 119-119.

R. Blanco, J.Tuya and B. Adenso-Díaz (2009). "Automated test data generation using scatter-search approach", Information and Software technology, vol. 51, Issue 4, pp. 708-720.

Remco R. Bouckaert(2009). "Weka Manual 3-6-1", Software manual, June 4, pp. 11-14.

Sinha A., and C.S. Smidts (2005). Domain specific test case generation using higher ordered typed languages from specification. Ph.D. Thesis, University of Maryland.

Subraya, B. M. and S. V. Subrahmanya (2000). Object driven performance testing in Web applications. Proceedings of the First Asia-Pacific Conference on Quality Software, Oct. 30-31, Hong Kong, China, pp. 17-26.

T. Y. Chen and M. F. Lau (1998). A new heuristic for test suite reduction. Information and Software Technology, 40(5): pp. 347-354.

Tapas Kanugo and David M. Mount (2002). "A local search approximation algorithm for K-means clustering" Communications of ACM.

Yanping Chen and Robert L. Probert (2007). "Regression test suite reduction using extended dependence analysis" Communications of ACM.

Corresponding Author

Khadija Sania Ahmad*

M. Tech in CSE Scholar, Al-Falah University, Dhauj, Faridabad, Haryana

E-Mail – ahmadsania18@gmail.com