University of Sindh Journal of Information and Communication Technology (USJICT)
Volume 3, Issue 3, July 2019
ISSN-E: 2523-1235, ISSN-P: 2521-5582
Website: http://sujo.usindh.edu.pk/index.php/USJICT/

Systems Development Life Cycle Test Driven Technique and Defect investigation

Soobia Saeed[1], Asadullah Shaikh[2], Muhammad Ali Memon[3], Muhammad Ali Nizamani[3]
[1]Department of Software Engineering, University Teknologi Malaysia,
[2]College of Computer Science and Information Systems, Najran University Saudi Arabia,
[3]Institute of Information & Communication Technology IICT University of Sindh,
soobiasaeed1@gmail.com, asshaikh@nu.edu.sa, muhammad.ali@usindh.edu.pk, ma.nizamani@usindh.edu.pk

Abstract: Software testing is essential to build and improve programming. Software testing is the basic order of construction software. Testing is undoubtedly expensive and involves the biggest individual time. Its supreme undertakings to conquer the costs and improve the viability of decreasing software implementation and expand the hazardous aspects of the quality level. The examination explores the convergence of test schemes with the counteractive action of imperfection and the following measurements used in Software Improvement by which the execution and viability of the value product is achieved. The research study further emphasizes pre-testing and quantifiable measurements of execution by which testing efforts can be broken down and the result can anticipate software execution within the progress process. The research study provides information of interest to understand and apply test techniques as desired. It is essential to make the correct decisions to embrace test techniques that fit the prerequisite of the venture. This study examines what works best together with SDLC and provides center information on when and how best to apply them. The measurements decide the decreased deformity rates to enhance execution by comparing each other's imperfection rates for each module or portion of usefulness. This assigns more assets and time before organization to settle deformities, resulting in a reduced rate of imperfection after sending reported by end customers.

Keywords: SDLC, software, testing systems, clients

I. INTRODUCTION

Information technology becomes quicker and changing quickly in every day, in this inciting development and substantial foundation control is currently finished with the assistance of developing and conjuring new procedures and applying them and upholding the group to execute these methods as day by day and rehashed errands to meet client prerequisites and expanding trust and business without a moment's delay by giving the privilege tried and bug free item. Additionally estimation is the vital to examine and figure execution through testing measurements. It gives speediest reaction, adaptable and savvy results [1].

A. Objective

The objective of research is to study and analyze the following key factors:

- Study on test rates and profound kind
- Provide point of interest and first-hand information in one document for apprentices to start testing the actualities and processes of the world.
- Measurement of the use of measurements and their viability to predict programming progress Study and investigate fundamental SDLC stages to adjust powerful

B. Comprehensive Study on Software Testing

Testing is a motion for QA. Testing is conducted as a distinct and complete phase in routine advancement methodology. Both robotic and manual test systems are a day by now. The programming product is becoming complicated and because it requires enormous manual frameworks to be robotized, quality issues and testing are becoming strict [3].

C. Software Testing Method

The effective software approach comprises of test instances intended to form a sequence of developmental steps that lead to a successful product. Strategies for testing provide a road map for testing. It is the management team's duty to create flexible software policy that can be modified to pick the correct route. There are four basic types of software testing, identified as follows, which plays a critical role in the stage of growth.

D. Component Testing

Developers ensure that each system created is bug-free before forwarding it to testing for the workers. In the design stage, developers ensure device accidents based on basic requirements and source code errors are removed. Therefore, the exam team can check the customer's viewpoint.

E. Assimilation Testing

After successful unit testing, all sections are merged as well as next levels are tested for integration testing. Integration testing consists of two classifications Underside up integration testing and Top Straight up integration testing. The original type Bottom-up usually starts with unit testing the reduced level than the higher level I at first this particular. Age Modules and shapes. Top down will be pursued next, generally starting with top-level web theme testing and gradually as well as lower-level web theme testing.

F. Top down Assimilation

The incremental strategy is to generate organization of the programmed. The integration of the module is performed by moving downwards, beginning with the fundamental control module.

G. Bottom up Assimilation

Starting as the name suggests the approach from the individual modules. The components are integrated from the bottom up; the required processing is connected to the integrated module, reducing the need for stubs.

H. System Testing

This test is carried out by the exclusive team when almost all parts are incorporated in order to ensure that certain quality standards are thoroughly tested. System testing is important because it is the first test in software development, that implementation is attempted in its entirety indicating that the software is directed towards the deployed setting, which is why the system can be produced. It allows the specific testers to check and check the earlier defined standard standards for that user expectation and company requirements compliance along with application specifications. System testing has any major kinds that can be Recovery Testing, Protection Testing, Graphical Program Testing, & Research Compatibility.

I. Acceptance or User Acceptance testing

The quality assurance team carries out acceptance testing to guarantee that the system works in manufacturing as required and user expectations are defined. Not only is it performed to check spelling, cosmetic flaws or not only interface problems, it also drags out potential mistakes that can crash the application system. It usually comprises of client testing and recognition on the grounds of the implementation created by other companies according to their demands. From the inner scheme, the customer is not much known, so it comes under the black box test method. This test also called testing for quality assurance, testing for validation, final testing and testing for implementation.

II. SOFTWARE TESTING TYPES

A. Regression Testing

Regression testing is done to confirm this shift after bug fixes have never impacted the system's other features. Due to its gap finding in implementation after modifications made, the significance of regression testing is owing. The verification of the other parts of the application probably did not affect the adjustments made to fixes. When software is regressively tested, mitigate raises the risk factor. During regression testing, the time lines are useless because they are quick and concise to verification norms

.

B. Alpha Testing

Whenever unit, incorporation and framework testing is consolidated called Alpha testing. It is performed by the two engineers and QA groups.

C. Beta Testing

When alpha testing is performed, beta testing is conducted effectively. It is also referred to as Pre-release. The application is mounted to execute and use the application before the real launch for the broad spectrum of test customers. User will install, operate and provide feedback. Fixes will be produced before the real launch during beta testing if bugs are reported by the test customer. As more bugs are recorded, the high quality is increased. Having greater quality improves the organization and business ' customer satisfaction and development.

D. Performance Testing

This test is recognized in terms of velocity, ability and stability as the significant and compulsory testing in software.

• Load Testing performs testing based on big information input and peak user and information load. Most of the load testing is done by automated instruments such as Load runner, Apache JMeter, AppLoader, IBM Rational Performance Tester.

• Stress testing — Check behavior in unusual circumstances such as shutting down and restarting network ports. Turn on or off the databases. CPU, memory, consumption of server resources by distinct procedures.

E. Usability Testing

Software system user friendliness testing defines the precise user behavior to be conducted on the application in order to fulfill their expectations. It is used to detect mistakes merely through user view, activities and use because of its black box testing. In five components I Nielsen clarifies usability. e. People with efficiency, errors / safety, ability to memorize, satisfaction, and

capacity to learn. Nigel and Macleod indicated how it was the requirement quality measure tested on the grounds of software results. In addition, ISO-9241-11, ISO-13407, ISO-9126 AND IEEE std, as well as quality models. 61012 In terms of characteristics and their sub-properties, describe this test.

F. User Interface Testing

It's the usability testing subpart. It is made up of GUI testing. Ensures the necessary elements such as color, alignment, size and other characteristics.

G. Security Testing

Ensure the safety of the software against vulnerabilities, whether known or unknown. Security testing should guarantee the primary elements such as integrity, accessibility, permission, authentication, data security, safe rules and regulations implemented injection faults, problems with session management and buffer overflow problems.

H. Portability Testing

It is possible to move the reusable and software from one location to another [22]. Building executable (.exe) to test software on other systems and second installing the application from one machine to another is the fundamental techniques used for portability testing. The main focus of this form of testing involves general testing of distinct environments through its use, computer hardware, operating system and testing of browsers. The prerequisites for this test must be to ensure that the testing of the unit and integration has been carried out. The software design should be based on the application flexibility and its portability requirements. To ensure the test environment will be established easily anywhere [4].

I. Testing disciplines

1. Specification based testing

It is possible to publish the test specification in either for mal or non-
formal language. There are three kinds of specifications: p rocess algebras, describe system conditions for interaction and active agent behavior; algebraic specifications, outlin ed in terms of system operations applications; Model-based specification, build explicit system state models and execute how to change the different activities.

2. Random testing

Test instances are selected randomly. Because of its efficient outcomes and prompt finding of failures, it is well recognized and adaptive among users. Whether, there is no option for the test developer to select event in sequences. It can lead to big numbers of test instances and resource wastage owing to unworkable test instances. [17]

3. Test Metrics

Software bugs cost the bulk of the productivity of the software. Identifying and removing early defects policies can lead to reduced defect rates. It is often said that what you can't measure can't be improved [15]. The standards of measurement are known as test metrics. Estimate software development activities ' usefulness and efficiency. Throughout the test attempt, they are gathered and constructed. Provide an objective measure of a software project's achievement. Test metric comprises of a few easy steps for quality outcomes and helpful measures i.e. Keep it easy, make it meaningful, follow it, and use it [21].

4. Deficiency

Deficiency is something that requires suitable and concentrated testing to remove from code. Defect is anything that the software dose does not mention but reflects software. Defect is the unwanted result of the software system [16]. During the development cycle, there are two kinds of flaw management to regulate flaw frequency in a documented manner to maintain track and record for the future. These are:

Deficiency detection

III. RESEARCH DESIGN

The assessment is carried out in the type of exploratory research. This sort of studies is generally aimed at gaining and locating thoughts and insights about the chosen subject [5].

Research is dependent on secondary funds, event surveys and knowledge. Detailed research of the SDLC phases and their measurement significance respected the difference in defect reduction before and after deployment of the software product. The study could concentrate on your literature and research findings and findings. Secondary research sources include study articles, publications, and separate observations of consultants, review and evaluation of literature. Literature search and case analysis based on observed results during SDLC focused primarily on evaluating and reporting the efficacy of defects with adequate follow-up testing and tracking of the development cycle. This research includes multiple data resource analyzes and data and information observations from both personal and external resources. This research study sample is component of the bug folder that was developed throughout the development stage of the project. To evaluate the cost and effectiveness of the test techniques used, the technique of defect evaluation and error prevention is used. Exploring techniques for over a year and undergoing latest and formal methods of testing implementing improvement and more domain exploration of test techniques.

A. Collection of Data

The data collected for assessment and findings will be based on the extensive life cycle of software development and the process analysis of defects will be discussed on. The basis of available data, irritating files and outcomes.

B. Methodology

In this fast-growing globe of software engineering, due to the elevated expectations of the user and notable quality products, as it is not simple. Indeed, it would be hard to properly grip the project at this stage of globalization, finish it on demand and enable it to be effective from the restricted funds on time. The research assessment is carried out on such a statistical basis as project experience and results to narrow down the characteristics to be regarded for the high quality and achievement of the challenge Adaptive strategy has been developed in software engineering to implement trending testing means to improve software testing and efficient outcomes to produce quality products. Effective techniques include Test Driven Development (TDD). Requirements serve as the basis for generating amount of test instances for software parts to detect bugs and mistakes from each component's information structures and functionality. TDD emphasizes the idea of launching a test plan before the software architecture is developed and continuously monitoring and monitoring defects [11]. The vital enhancement of SDLC-focused characteristics combined with TDD method is used to solve the SDLC defect rate merely by highlighting the following significant strategies:

- Constraint gathering and analysis
- Development
- Communication
- Define Process
- Skills and knowledge
- Tools
- Panel work
- Craving to give something extra

IV. CONSTRAINT GATHERING AND ANALYSIS

The first stage in the creation of software is to comprehend and list the demands according to user and client needs. Requirement should be measurable from the beginning of the SDLC and track able. This allows the quality culture from the start of the creation of software. For such outcomes, the test oriented method is appropriate. Each requirement should have at least one or more test cases with distinctive traceability identification. Analyzing and measuring, selecting and planning metrics to assess requirement modifications as follows:

A. Requirement unpredictability

Total # of requirements v/s total # of changed requirements

= (# of Requirements Added + Removed + Changed) *100

Total # of Requirements

Makes sure the requirements are defined correctly while estimating

Here, Total req. = 67 in start, further digging into the project requirement phase added new req. = 7 and afterwards deleted some requirements = 3 and changed requirements = 11

So, it can be estimated as

= (7 + 3 + 11) * 100 = 31.34%

67

The results shows 1/3 of the requirement changed after initial requirement gathering

B. Development

Preparing is the basis of nearly any successful project. It includes the consideration and execution of all significant operations planning throughout the design phase and also after deployment. The design stage includes the potential for weakness, strength and team, the availability of instruments, requirements, limitations and methods to address barriers. The result of excellent coordination and participation of suggested subsidiaries plus client inputs or even stories will be the excellent growth. More impressive range is performed and enforced at all levels. The TDD approach can be used to reduce defects by analyzing and controlling defects throughout the requirement and development phase by testing the generated test cases. To carry out planned unit testing, the test cases are distributed to the team and the system is also being tested. This procedure diminishes the genuine deformity rate during the whole improvement and test stage. These pre-produced test examples as of now guarantee central and useful testing. The bud record demonstrates the details saw during the trials on finishing of one component during the development. The exploration demonstrates the outcomes as deformity driven examinations by the measurement utilized.

Effectiveness of Defect Removal

DRE = (# of Defects fixed during development cycle x100%) / Defects latent

Defects latent = Defects fixed within development cycle + defects reported and found after deployment. Here, observed and recorded data shows the following statistics

Defects found within the development cycle = 68

Defects reported by users = 13

Defect latent = 68+13= 81

DRE = (68*100)/81 = 83.9%
The result shows the reduction in reported defects after deployment by users. The major part is covered in development cycle by analyzing daily defect prevention and fixing efficiently with pre-testing technique.

C. Communication
In the SDLC, communication plays a crucial role. Whether its team cooperation, management cooperation and most importantly client communication is proving efficient and different outcomes with a strong comprehension of needs. Due to time and effort concerns, it is observed as the hardest job. While communicating as much as necessary, the client may demonstrate less interest. The team must use various methods to capture customer care and their quality time to reduce project flaws. Here a need for adequate network communication or alternatives is required to discuss project procedures. Best practices may include casual and scheduled meetings, surveys, prototype, sample verifications, confirmation and reminder calls and emails, and many other social methods of adopting cooperation for best and enhancement. Accomplishment of Deficiency reduction process within SDLC.

The project cannot succeed without having to pursue any practice. From the planning stage the defining mission is certainly the efficient approach to completing the project irrespective of the planning is to follow agile or even waterfall processes. However, the method must be described to measure and schedule the real cycle and subsequent implementation of the defect decrease. The initially proposed task is the requirement for measurement and tracking and the rate of defects across the SDLC [19]. Generate hand-to-hand documents for the performance of the project. A bug record is retained and updated as the stage of product development to make this occur. As a result of analyzing the element-based performance, the observed flaws were listed. By using this approach, the method demonstrates the notable difference in the decrease of previous and present faults.

D. Software Tools
The decision to apply suitable and space explicit instruments improves the proficiency of the venture. Devices can diminish your time and less asset cooperation with profitability and detectability contrasts. Testing includes the ongoing patterns and focused results to improve the nature of the item. Testing endeavors can be decreased by utilizing devices, as testing is without a doubt an exorbitant stage. The remarkable yield may change in necessity following, bug following, announcing instruments, and the board instruments.

E. Panel Work
The co-individuals and cooperative people can all the more likely understand certain conditions and think of phenomenal contemplations over the encounters part. This is the minute to value every part's view and proposals in the advancement and arranging stage to conquer the mistake rate. Any place there is a quality cooperation, the private sense of self spots. Worth the group and underscore the possibility that every single individual is in charge of venture achievement and quality control. It is the executives' obligation to give such air and inspiration to quality venture accomplishment.

F. Craving to give something extra
The seasoned saying is that giving clients some extra time to develop and deploy earns respect and satisfaction. In trust development, helpful data supplied by the business during non-working hours or little care to avoid critical flaws can create a difference.

G. Charts
Bug reporting on one-week leadership and fundamental module. Upon completion of each test cycle, progress towards leadership is recorded to obtain a clear view of the quality of growth as well as the occurrence of defects and their fixes.

TABLE.1: Metrics Calculation Data

Base Metrics	Calculated Metrics		
Metric	Value	Metric	Value
Total # of TCs	100	% Complete	47.0%
# Executed	64	% Test Coverage	64.0%
# Passed	47	% TCs Passed	73.4%
# Failed	5	% TCs Blocked	8.0%
# UI	4	% 1st Run Failures	15.6%
# Blocked	8	% Failures	20.3%
# Unexecuted	36	% Defects Corrected	66.7%
# Re-executed	5		
Total Executions	74		
Total Passes	47		
Total Failures	15		
1st Run Failures	10		

TABLE.2: DEFECT TRACKING GRAPH

Actual Results		
	New	Resolved
Day	Defects	Defects
1	1	0
2	2	1
3	5	1
4	10	7
5	8	2
6	6	18
7	12	1
8	9	8
9	14	12
10	9	6
11	5	15
12	8	12
13	9	13
Total	98	

V. RESULT & DISCUSSION

After studying software evaluation methods and strategies, software testing could be an ongoing and coexisting development practice. We should talk about the critical components of testing that could be helpful and reasonable during the existence cycle of programming improvement. Necessity designing could be the strategy to evaluate, take and keep up programming prerequisites. Counting the Realistic Unified Process [7], Extreme Programming [8] in conjunction with Scrum [9] means that requirement engineering is often an continuing activity in your project development lifetime.

Without specifying the prerequisite, testing is impossible. All requirements of linked test instances alone must be traceable and measurable. Each requirement should have ID relevant for testing ID and at least one requirement ID should be associated with each test ID[10].Testing is often the team's heavy and accountable work. Developer must have separate vision and scheduled testing specifications to ensure that the tester team does not receive any buggy code. Next, if testers participated during development using the ongoing testing operations along with all test instances produced prior to a finished development stage, designers will readily hide and overlook bugs during unit evaluation. As a result, team players from the specified moment spam are testing and verifying specifications every moment in all procedures.

Unfortunately, testing and testers have been noted to finish their assignments in brief spam time. As it took a lot more time in actual growth and it is too brief to conduct all experiments in less time for testing. In this scenario, pre-planning and defining functional specification can solve the time issue. Upon completion of the design phase, the testing method is also prepared to carry out all functional tests to test the error rate. The benefit of a functional specification is that the generation of test instances is also began in parallel during the development phase, so that when the production code is prepared for test instances, the sequence of bottlenecks is removed from the development process. Secondly, it is recorded so that outcomes and observations can be analyzed by reporting defects, keeping track of the performance of each finished assignment, fixing defects on time within the development period and involving expert opinion by reporting defects and effectiveness metrics to highlight the weakness and conduct evaluation to enhance and reduce the rate of defects within the development process. Sharing and listening policy with clients to clarify stuff from the user's point of view can readily confirm that the build item meets anticipated demands. This helps to very carefully comprehend the demands and makes it easy to map and schedule the process to satisfy the functional requirements.

VI. CONCLUSION

The measurements displayed for estimating testing endeavors give proficient a noteworthy advantage, perceivability in the accuracy of the improvement and status for discharge or generation, and the product item standard a work in progress can be watched. This offers the key data for making choices to deploy along with stopping testing attempts or even improving efficiency in the missing region with good measurements to finish the quality product well-timed and decreased error rate of around 33 percent by comparing a few performance modules one by defect-driven reporting with another by easy software.

A. Future Work

Software testing is costly and SDLC operation takes time. Recently, automation is the main factor that reduces many researchers ' testing attempts. The future research will be based on the finest instruments to automate the testing process with cost and effort decrease.

B. Presentation Analysis

The last presentation involves an overview, study methodology and research-related objects. This exploration study offers a reasonable picture of how to keep up quality from the earliest starting point of the assembling connected to the product framework and precisely what is the hugeness of programming testing in the consistently finishing programming item advancement life cycle. The deformity driven strategy is utilized to reveal insight into primary concerns of SDLC techniques to diminish the mistake rate in the SDLC and after arrangement.

ACKNOWLEDGEMENT

Authors are grateful to the Department of Software Engineering, UTM, to carry out this work.

REFERENCE

[1]. Tosun, A., Dieste, O., Fucci, D. et al. Empir Software Eng (2016), An industry experiment on the effects of test-driven development on external quality and productivity doi:10.1007/s10664-016-9490-0

[2]. Davide Fucci , Burak Turhan , Natalia Juristo , Oscar Dieste , Ayse Tosun-Misirli , Markku Oivo, Towards an operationalization of test-driven development skills, Information and Software Technology, v.68 n.C, p.82-97, December 2015 [doi>10.1016/j.infsof.2015.08.004]

[3]. Davide Fucci, Giuseppe Scanniello, Simone Romano, Martin Shepperd, Boyce Sigweni, Fernando Uyaguari, Burak Turhan, Natalia Juristo, Markku Oivo "An External Replication on the Effects of Test-driven Development Using a Multi-site Blind Analysis Approach" in ESEM '16 Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, (2016)

[4]. Romano, S., Fucci, D.D., Scanniello, G., Turhan, B. and Juristo, N., 2016. Results from an ethnographically-informed study in the context of test-driven development (No. e1864v1). PeerJ Preprints.

[5]. Arthur, J.D. and Dabney. J.B.: Applying standard independent verification and validation (IV&V) techniques within an Agile framework: Is there a compatibility issue?. Proceedings of Systems Conference, IEEE, 2017.

[6]. Fleming, C.: Safety-driven early concept analysis and development. Dissertation. Massachusetts Institute of Technology, 2015.

[7]. Wang, Y. and Wagner, S.: Toward integrating a system theoretic safety analysis in an agile development process. Proceedings of Software Engineering, Workshop on Continuous Software Engineering, 2016.

[8]. Leveson, N.: Engineering a safer world: Systems thinking applied to safety. MIT Press, 2011.

[9]. Wang, Y. and Wagner, S.: Towards applying a safety analysis and verification method based on STPA to agile software development. IEEE/ACM International Workshop on Continuous Software Evolution and Delivery, IEEE, 2016.

[10]. Martins, L.E. and Gorschek, T.: Requirements engineering for safety-critical systems: Overview and challenges. IEEE Software, 2017.

[11]. Vuori, M.: Agile development of safety-critical software. Tampere University of Technology. Department of Software Systems, 2011.

[12]. Fucci, D. et al.: A dissection of test-driven development: Does it really matter to test-first or to test-last. IEEE Transactions on Software Engineering 43.7 (2017): 597-614.

[13]. Silva, T.R., Hak, J.L. and Winckler, M.: A behavior-based ontology for supporting automated assessment of interactive systems. Proceedings of the 11th International Conference on Semantic Computing. IEEE, 2017.

[14]. Falessi, D. et al.: Empirical software engineering experts on the use of students and professionals in experiments. Empirical Software Engineering 23.1 (2018): 452-489.

[15]. Enoiu, E.P. et al.: A controlled experiment in testing of safety-critical embedded software. Proceedings of the International Conference on Software Testing, Verification and Validation. IEEE, 2016.

[16]. Scanniello, G. et al.: Students' and professionals' perceptions of test-driven development: a focus group study. Proceedings of the 31st Annual Symposium on Applied Computing. ACM, 2016.

[17]. Kitchenham, B. et al.: Robust statistical methods for empirical software engineering. Empirical Software Engineering 22.2 (2017): 579-630.

[18]. Krenn, W., R. Schlick, S. Tiran, B. Aichernig, E. Jobstl and H. Brandl, 2015. Momut: UML model-based mutation testing for UML. Proceedings of the IEEE 8th International Conference on Software Testing, Verification and Validation, April 13-17, 2015, Graz, Austria, pp: 1-8.

[19]. Graf-Brill, A. and H. Hermanns, 2017. Model-Based Testing for Asynchronous Systems. In: Critical Systems: Formal Methods and Automated Verification, Petrucci, L., C. Seceleanu and A. Cavalcanti (Eds.). Springer, Cham, ISBN: 978-3-319-67113-0, pp: 66-82.

[20]. Lindvall, M., D. Ganesan, R. Ardal and R.E. Wiegand, 2015. Metamorphic model-based testing applied on NASA DAT: An experience report. Proceedings of the 37th International Conference on Software Engineering-Volume 2, May 16-24, 2015, Florence, Italy, pp: 129-138.

[21]. S Saeed, A. Shaikh, M.A. Memon, M.A.Nizamani, Faheem Ahmed Abbasi, Syed Mehmood Raza Naqvi, " Evaluating the Quality of Point of Sale (POS) Software". University of Sindh Journal of Information and Communication Technology (USJICT), Volume 3, Issue 2, April2019.

[22]. S.Ali, M.A.Memon, K.T.Pathan, F.A.Abbasi, " Comparative Analysis of Location Based Technologies Inorder To Develop IOT Applications". University of Sindh Journal of Information and Communication Technology (USJICT), Volume 3, Issue 2, April2019.