Knowledge-based Acceptance Test driven agile Approach for Quality Software Development

N. Asha, Prasanna Mani

Abstract: ***Agile approaches in DevOps context evolved SDLC to focus much on iteratively, communication and interactivity between different project roles. In this service-oriented world, much attention has to be given to customer specifications by providing continuous applications delivery with reduced time-to-market. Testing and the Quality Assurance (QA) activities take a central role in ensuring the accomplishment of users' acceptance criteria and the quality of the delivered software. Instead of contractual approach the agile approach is used to emphasis on taming the Requirements Engineering, Testing and Quality Assurance activities. This increased importance of testing methodology manifests the software developing companies to advance further on testing approaches, preventing defects during the development process. This paper presents a testing methodology to apply Acceptance Test Driven Development (ATDD) techniques while developing DEVOps projects, termed Acceptance Test Simple Testing (ACT-ST) methodology. ACT-ST approach is very evident, supported by the open source framework that generates test cases using the syntactic structure of Gherkin language from ATDD scenario specifications extracted from the user stories quoted with acceptance criteria. ACT-ST approach promotes continuous Metric-based Quality Check, structured User Stories with acceptance criteria for test case generation, agile test reporting, Knowledge Repository and Functional Knowledge Documentation for governance of Quality Management System (QMS).***

Keywords: Software testing, ATDD, Quality Management System, DevOps.

I. INTRODUCTION

An underlying assumption in plan-driven methods is that the requirements are relatively static. On the other hand, iterative method like agile approaches relies on change and diagnose that the only constant is change. Agile approaches help teams to respond to unpredictability through incremental, iterative work cadences and empirical feedback. Agilest put forward alternatives to waterfall, or traditional sequential development. Agile is termed as umbrella term that includes methodologies like Scrum, XP, Crystal, FDD, and DSDM. Agile provides better way to value: Individuals and interactions over processes and tools, working software over comprehensive documentation, Customer collaboration over contract negotiation, responding to change over following a plan.

A recent research survey result shown in Figure. 1, conducted by Hoa Loranger and Page Laubheimer of NN/g Nielsen Norman Group in U.S., U.K., Australia, Singapore, had a combination of surveys with in-depth interviews and case studies. Overall, 356 respondents (UX practitioners) participated in this survey with around 24 professionals were part of in-depth interviews and case studies. The study reports that 69% of the projects they support are agile, this is a significant increase compared to the report that was obtained in the year 2008 which was 47%.

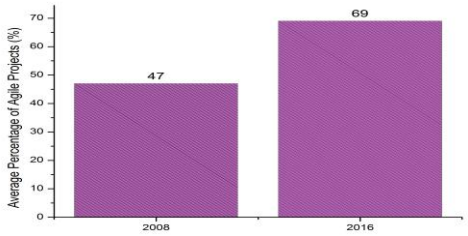

Figure 1: Agile adoption is growing

"More than the act of testing, the act of designing tests is one of the best bug preventers known" – Boris Beizer. According to agile manifesto principles, "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software." Precisely agile approaches endorse reducing the time-to-market to incorporate continuous delivery, so the central role called agile requirement engineering, testing practices has to be focused. One of the vital actions to be done in testing practice is; to construct Metric-based Quality Gates regularly to measure and validate the pieces of software continuously.

In simple words, speed needs a balance with quality checks. Agile implies divide-and-conquer strategy in its sprint concept to rethink how to address testing in comparison with traditional testing practice. Testing processes should adapt to iterative context and also the agile validation checks.

To accumulate knowledge in each iteration, we need to have a study view of the big picture of the software being delivered.

Based on this gained knowledge Metric-based Quality Gate fixed for Quality Checks. Testing role should be considered as an active part of the software engineering process to address all these agile challenges, to bring neat communication strategies.

This ACT-ST approach follows the agile principle *"Business people and developers must work together daily* throughout *the project"*, testing and QA roles can be included too.

This approach accustom new method of communication than seeing that "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation".

In many project contexts with distributed teams, variable people availability, integrate a neat complementary written communication artifact to reduce ambiguity and requirements misunderstandings to avoid conflicts with economic impact.

Revised Manuscript Received on December 30, 2018.

N. Asha, School of Information Technology and Engineering, Vellore Institute of Technology, Vellore (Tamil Nadu), India

Prasanna Mani, School of Information Technology and Engineering, Vellore Institute of Technology, Vellore (Tamil Nadu), India.

User Stories are commonly accepted agile requirements artifacts. User Stories supplemented with acceptance criteria (States the Conditions of Satisfaction) denotes the descriptions of a feature told under users' perspectives who desires the new capability. User Stories acts as a base for explicit communication among stakeholders, designers, developers and testers to generate accurate test case design in which agile iteration.

The ACT-ST approach introduces three active roles: Collaborative Manager, Test Reporter and Quality-Checker to narrow down the testing process. The ACT-ST approach is a Knowledge-based framework formalized with the support of knowledge gained from the approach introduced by Albert Tort.

ACT-ST promotes continuous Metric-based Quality Check, structured User Stories with acceptance criteria for test case generation, agile test reporting, Knowledge Repository and Functional Knowledge Documentation for governance of Quality Management System (QMS). The remainder of this paper is systematized as follows. Section 2 presents the related work made for test case generation. Section 3 describes the proposed framework and describes the main components of ACT-ST approach. Section 4 discusses work part of Knowledge driven component. Section 5 represents the algorithmic steps for test case generation.

Section 6 discusses about the functional model and functional document generation. Section 7 represents the result of proposed approach.

Section 8 compares the result of ACT-ST approach with three different agile project specifications: 1) Online EB payment portal, 2) Online shopping system, 3) Net banking system. Section 9 concludes the paper by summarizing the benefits of proposed approach and presents direction for future works.

II. RELATED WORK

The contractual approach was used so long; hence it has to be refined to address the current agile testing. At the limit Martin and Melnik says that tests and requirements are equivalent.

The agile movement has proposed to focus on User requirements in order to achieve quality in the testing process, followed by that Agile Acceptance Testing is proposed to bridge the gap between the developers and stakeholders. The user scenarios were expressed in different form: ATDD with Fit test framework, Cucumber. This how the requirements are expressed as acceptance tests and these tests are automated.

The research works mentioned in Table. 1 describes various approaches for test case generation in different context.

The proposed framework is a form of agile approach developed for DevOps environment and deals with both internal and external testing.

External testing produces Acceptance Tests for being validated by project stakeholders.

Table 1: Different approaches for Test Case Generation in various contexts

Author(s) / Year	Proposed Framework	Capabilities	Outcome
Hong, Weiyin, et al., (2011)	Research Model: Tripartite model of attitude	The model incorporates an individual characteristic variable that is particularly relevant to individual's reaction to changes.	Developed research model by identifying key constructs under three dimensions (cognitive, affective and Behavioural) of attitude to derive hypothesis for adopting agile IS
Ashar Fuadi et al., (2015)	Tcframe	Single self-validating generator program for generating test cases, both test case generation and validation is done (runner program)	Enable to create test case more collaboratively
Carrera, Álvaro et al., (2014)	BEAST Testing Methodology	BEAST is an open source framework that generates test cases skeletons from BDD scenarios specifications , transparent traceability from user requirements to test cases.	Facilitates the communication between stakeholders, designers, developers and testers. Reduced test implementation time, transparency in translation. This tool is Validated for fault diagnosis.
Janus, André, et al., (2012)	3C Approach	Adds continuous measurement and continuous improvement as subsequent activities to CI in an agile practice	Enhanced automated compile, build and running of test. Established Metric-based Quality-Gates for an agile quality assurance.
Albert Tort et al., (2011)	TDCM Approach: conceptual modelling	Increases the semantic quality by evolving the set of test cases refined to give pass verdict. TDCM can be adapted to other languages.	Complete and correct schema at each iteration: After each iteration, the result is a correct and complete conceptual schema according to the processed test cases.

III. THE ACT-ST APPROACH

The ACT-ST approach focuses on bringing effective and efficient tactic for simplifying the test case generation. Additionally ACT-ST is a Quality-focused approach for enhancing agile activities in the development environment. This approach concentrates on four main objectives: (1) Input enriched semiformal User Stories with their acceptance criteria, (2) smoothen communication among stakeholders, developers, testers even they are located apart, (3) effective Quality Checks (Validation checks) across the agile process, and (4) enhancing Quality Management System (QMS) based on knowledge perspective in agile iterations by sustaining the big picture of the project. Overview of main components of the ACT-ST approach is exemplified in Figure. 2.

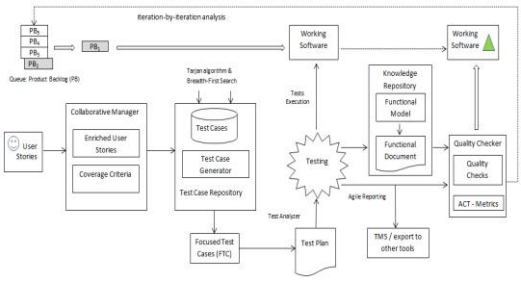

Figure 2: Overview of ACT-ST approach

A. Enhancing User Stories with Structured format

The agile approach entirely works on the customers' perspectives, in each agile iteration the practitioners' strives to satisfy the users. Hence our ACT-ST approach at its first stage completely concentrates on the Users Stories to understand the whole expectation of them on the agreed project. For better understanding and to avoid ambiguities User Stories is written in well-structured format using K+ editor / Gherkin-Cumcumber, which clearly formats the users' needs, their Acceptance Criteria and the configured level of Coverage Criteria.

B. Agility in Test Report

As a resultant of test case execution we get the test verdict that can be pass, fail or error. The test cases can be executed manually or it can be automated with the help of automated tools depending upon the project size, characteristics and project objectives. In agile practice the build of software is of iterative in manner; the testing process has to be repeated for every iteration in order to make the validation, integration and Quality Checks. This approach includes a smart trivial procedure to report about the focused test cases that have to be executed in case the agile context does not use a full specific Test Management System (TMS). This trivial procedure simplifies the test reporting effort in agile teams. The ACT-ST approach also includes the compatibility functions to report the test verdict to the Test Management Systems (TMS), Test Link, Microsoft Excel and also can be integrated with other tools for fine checks.

C. Functional Knowledge generation

The functional documentation is codified from the set of test cases generated from the enriched User Stories written in structured format. This Functional Document covers the complete specifications of the big picture of the project termed as Functional Knowledge. The Functional Document/Knowledge represents the detailed concepts, relationships and functional operations of each user defined functions. In analogous to this the Functional Model is depicted neatly that visualizes the operational mechanism and the relationships among the functions and their collaborations. In each agile iteration the updated new version of Functional Documentation and the Functional Model is generated. This Functional Knowledge acts as: 1) a base for extracting enriched system knowledge, and 2) a base for computing ACT-Metrics.

D. ACT-Metrics as Quality Checks.

ACT- Metrics is the dynamic component in the framework which comprise of a balanced dashboard for the active development of the project.

In each iteration the control panel indicates the metrics like the number of test cases, number of passing or failing test cases, etc.

Additionally, this component actively compares the knowledge data (Functional Knowledge) related to the number of concepts, relationships and functionalities that are covered by tests in every agile iteration for better Quality Checks.

IV. KNOWLEDGE DRIVEN COMPONENT

In this section we exemplify the Knowledge driven component that takes the enriched User Stories with priority indicators defined in flow sketches as input to generate: 1) Test designs (refer Table. 2) which include test case steps and assertions, 2) Filtered Test Cases for execution with available resources and stated Coverage Objectives.

The Test cases can be presented in HTML/Open Document file format and also be exported to other Test Management Systems, Microsoft Excel or Test Link.

Table 2: Generation of Test Case steps and assertions

Test ID	Objective	Preconditions	Steps	Checks	Priority
Add Beneficiary	Add beneficiary to the bank account	that a user is authenticated and check the beneficiary list before adding	1. Click on manage beneficiary 2. Click on Add beneficiary	The beneficiary is added to the bank account	Must
Fund Transfer	Transferring amount to other banks or own bank accounts	that a user is authenticated and make sure that amount should be available	1.click on fund transfer menu 2.click on own accounts if you are transferring to same account 3.click on other if you are transferring to other accounts	Is amount debited from account or not	Must

V. TEST CASE GENERATION

The ACT-ST approach incorporates the following stages to generate the test cases: 1) Eliciting end-to-end paths to derive all possible combinations, 2) Unit Test Case generation for each User Story with its acceptance criteria, 3) End-to-end Test Case generation for all possible combinations, 4) Combining Test Case with different Test data.

A. Eliciting end-to-end paths to derive all possible combinations

To compute the end-to-end paths for a general User Story the activity diagram is extracted from the User Story; this step of process can be omitted for atomic User Stories.

In this stage the two popular algorithms are used subsequently: Tarjan algorithm and the Breath-First Search algorithm.

The Tarjan algorithm: to identify strongly connected components to make a neat flow diagram avoiding infinite loops to ensure that one single traverse is made for each cycle while deriving the Test Case paths.

The Breath-First Search algorithm: it guarantees full coverage of all possible paths and the User scenario associated in each path to compute end-to-end Test Cases.

Algorithm 1: To identify strongly connected components	Algorithm 2: To identify all possible paths

function StrongConnect(vertex *u*)
num←*num*+1
//increment *num*
order(*u*)←*num* // set *order*(*u*) to smallest unused number
link(*u*)←*order*(*u*)
// least *order*(*v*) accessible is *u* itself
push *u* on *S*
for all neighbors *v* of *u* **do**
if *order*(*v*) is undefined **then**
// *v* has not been visited
StrongConnect(*v*)
link(*u*) ← min(*link*(*u*),*link*(*v*))
else if *v* is on stack *S* **then** // *v* is in current component
link(*u*)←min(*link*(*u*),*order*(*v*))
if *link*(*u*) = *order*(*u*) **then** // *u* is root of component, create SCC
create new strongly connected component
repeat
v ← top of *S*
add *v* to strongly connected component pop top from *S*
until *u*= *v*
function Tarjan(G(V, E))
num ← 0
initialize new empty stack *S*
for all vertices *v*∈ *V* **do**
if *order*(*v*) is undefined **then**
// *v* has not been visited
StrongConnect(*v*)
	BFS(G, s)
	for each vertex u ∈ V [G] −{ s } do
	state[u] = "undiscovered"
	p[u] = nil, i.e. no parent is in the BFS tree
	state[s] = "discovered"
	p[s] = nil
	Q = {s}
	While Q = Ø do
	u = dequeue[Q]
	process vertex u as desired
	for each v ∈ Adj[u] do
	proce ss edge (u, v) as desired
	if state[v] = "undiscovered" then
	state[v] = "discovered"
	p[v] = u
	enque ue[Q, v]
	state[u] = "processed"

B. Unit test case generation for each User Story with its acceptance criteria

The Unit Test Case is designed for each atomic User Story. *Given* clauses corresponds to Test Case preconditions, while *When* and *Then* clauses are essential to generate steps and Test Case assertions. Figure. 3 Shows the Unit Test Case for "Valid Login".

Identifier: TC001 Login to NetBanking _ValidLogin	
Test Objective: Valid login when login to the Net banking	
Preconditions: I am in the login Page	
Test design	
Test Steps	**Test Assertions**
1. I enter my username "tom" and password "tom123"	
	2. The system indicates am logged in

Figure 3: An example of Unit Test Case generated

C. End-to-end Test Case generation for all possible combinations

The paths found in the first stage and the Unit Test Cases designed in the second stage is fed as input for end-to-end Test Case generation. This stage combines all paths and covers all possible User scenarios to compute the end-to-end Test Cases, (In this stage Test Cases are generated commonly without including the example test data). The fragment is shown in Figure. 4.

Identifier: TC002_ Fund-Transfer_Valid Login	
Test Objective: Valid login, add beneficiary and transfer funds. **Preconditions:** Earlier I should login **Test Design**	
Test steps	**Test Assertions**
1. I enter my username "tom" and password "tom123"	
	2. The system indicates am logged in
3. Add beneficiary to the account	
	4. The system indicates that one beneficiary is added successfully.
5. Go to the fund Transfer page and select inter/intra beneficiary	
6. Enter amount to be transferred and give transaction password	
	7. System validates the debit account, transaction password and sends an OTP to registered mobile number
8. Enters the OTP	
	9. System verifies the OTP if correct it states that the amount transferred successfully else to try again.

Figure 4: End-to-end Test Case generation.

D. Combining Test Case with different Test Data

The acceptance criteria that stipulate example data is considered to draw input data set. The Test Cases extracted in previous stage is tested with this input test data; this stage combines the extracted test cases with induced test data for test execution.

E. Focused Test Cases

Using the above mentioned combination of algorithmic procedure the set of all possible Test Cases are generated. For a smart testing process it's not mandatory to execute all set of Test Cases and there is no sufficient time and resources to execute too, hence the Knowledge driven component lets the whole set of test cases to be filtered based on two attributes:

1. Test Case execution for specific User Stories: the User specified User Stories are targeted for testing and the Test Cases relevant to those User Stories are alone considered for the next level of filtering.
2. Test Case priority: based on the priorities set for each targeted User Story, the priority values (High, Medium, Low) is set for its associated each Test Case. The filtering of Test Cases is made based on this priority value set. The resultant Focused Test Cases alone is executed to save time and other resources.

$USS = \{ USS_1, USS_2 \ldots USS_n \}$ ◁ USS: User Specified Stories
$TTC = \{ TTC_1, TTC_2 \ldots TTC_n \}$ ◁ TTC: Total Test Cases
Step1: $P = USS \cap TTC$ ◁ extracts the Test cases of user specified (UTTC)
$Q = \{ UTTC_1, UTTC_2 \ldots UTTC_n \}$ ◁ UTTC: User Targeted Test Cases
$PS = \{ H, M, L \}$ ◁ PS: Priority Set – (priority value of test cases high, medium, low)
Step2: $F = \{ Q \} - \{ L, M \}$ ◁ extracts the Test cases of priority value: high
$F = \{ FTC_1, FTC_2 \ldots FTC_n \}$ ◁ FTC: Focused Test Cases

VI. ITERATION-BASED DOCUMENTATION GENERATION

The User Story is formalized further using a restricted syntax as shown in Figure. 5 to reduce the ambiguity, by doing so will support in smooth generation of Functional Model and Knowledge Documentation.

- Test-Driven Functional Model Generation: The living model (evolves as generated test cases changes) is obtained using the engine stated in this model (specified using UML) postulates the concepts, relationships and operations derived from use cases, which in turn is derived from User Stories. Figure. 8 shows the splinter of an output model of the knowledge related to Net Banking system. In agile iteration the model evolves as new knowledge is added. This model is a knowledge repository that represents the big picture of the project termed as functional knowledge components.
- Knowledge Document Generation: The model obtained can be structured into knowledge document that can be exported to several formats like HTML, PDF, and textual Open Document file.

```
Feature: [ Verb sentence]
   As a [Role name]
   So that [Verb sentence]

Scenario: [Free text]
   Given [GoTo(pageName) | objName is a ConceptName]
   When [Verb sentence with  #values and <parameters>] and ...
   Then [Boolean verb sentence]

Examples:
   | [parameterName]  |  [parameterName]  |
   | [value]          |  [value]          |
   | ...              |                  ...|
Example:
Feature: Log in to the Net Banking
As a User
I Want to login as Account  Holder
So that I want to save my transaction details

Scenario: Valid Login
Given Go To (login page)
When I enter my username "tom" and password "tom123"
Then the system indicates am logged in

Scenario Outline: Invalid login
Given Go To (login page)
When I enter an invalid <username> and < password>
Then the system indicates that the entered data is invalid.

Examples:
| username   |  password   |
| "jatsan"   |  "jat123"   |
| "moni"     |  "mon123"   |
```

Figure 5: Formal User Stories specification

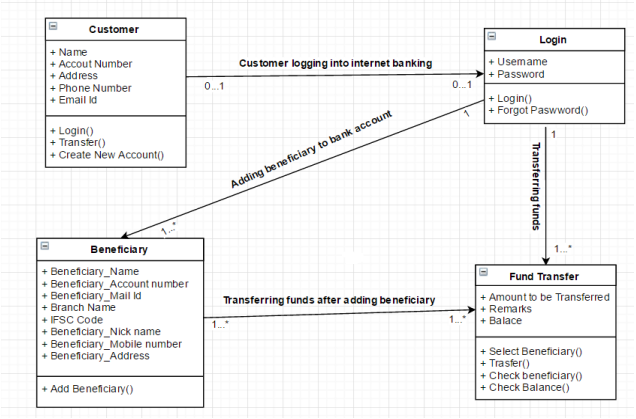

Figure 6: UML diagram fragment of Net Banking

VII. METRICS- BASED QUALITY CHECKS FOR AGILE MANAGEMENT

The project management dashboard is used to monitor and control the agile projects. It visualizes the outcome of each agile iteration and computes the knowledge metrics that takes into account the inputs like: User Stories with Acceptance Criteria, set of Test Cases generated, verdicts of each Test Cases, Functional Knowledge specified using UML. This computed innovative metrics provides more information for example: productivity of Test Cases, productivity of each agile iteration. Such metrics are compared amongst them to work for better outcome in the next iteration of testing activities.

VIII. RESULT

The proposed approach is well suitable for enhancing the agile activities in the context of DevOps environment. The ACT-ST approach comprises of many features that together offered many advantages to the agile environment: 1) it reduces the Test Case design time, since it is generated from enriched User Stories, 2) Functional Model is evolved easily through User Stories defined in semiformal syntax, 3) Test Case execution and Reporting is computed smartly to manage the system Quality, 4) Knowledge Documentation is generated from functional model in each agile iteration to derive Knowledge metrics, 5) Quality management dashboard is provided for balanced evolutions.

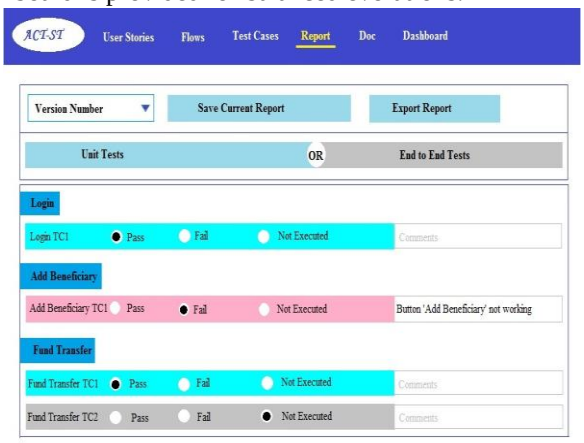

Figure 7: Test cases reporting interface

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

The interface of testing report is shown in Figure. 7. In each agile iteration the model is evolved by adding newly gained knowledge and the project management dashboard presents the graphical information of basic knowledge metrics which is shown in Figure. 8.

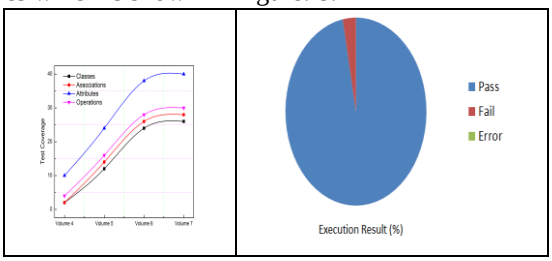

Figure 8: Basic Knowledge metrics

A. Comparative Results

The experiments were performed to determine that the ACT-ST approach has enhanced the agile activities to achieve reduced time-to-market and to fetch Focused Test Cases effectively. The proposed framework provides the possible Test Cases with 100% Coverage criteria. This paper compares the result of ACT-ST approach with three different agile project specifications: 1) Online EB payment portal, 2) Online shopping system, 3) Net banking system. The result shows that the semiformal User Stories based ACT-ST approach gives better performance than the previous approaches without User Stories and acceptance criteria. The proposed approach smartly filters the Test Cases based on User Specifications to reduce time-to-market time period. Figure. 9 mentioned below illustrate the graphical representation of the result obtained.

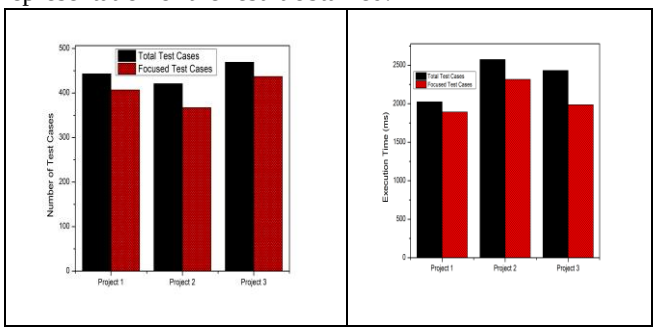

Figure 9: Performance of ACT-ST shown with three different agile projects

IX. CONCLUSION

The ACT-ST approach presented in this paper yields an effective solution for enhancing the agile iteration activities in the context of DevOps environment. The test case generation method used in this approach generates the test case based on users' perspective and provides complete coverage of user stories with acceptance criteria; the filtering method provides (focused test cases) a smart way to focus on the needed test cases and screens out the able test cases for DevOps projects. The generated test cases are exported to HTML/word processing file format or integrated to test management systems. This user stories based approach helps in improving the performance of testing process by applying the knowledge metrics obtained from functional knowledge components (new knowledge is added in each agile iteration to enlighten the testing activities), increasing the frequency of continuous delivery of application, reducing the time-to-market. It incorporates Tarjan algorithm and Breadth-First Search algorithm which renders 100% path coverage. This approach is implemented in three different agile projects which has given good optimized outcome and enhanced the testing activities. Overall this approach helps to face the challenges in maintaining good quality in the testing activities in the context of agile practices by reducing the test design effort and making use of knowledge components for innovative management.

REFERENCES

1. "Accounting for UX work with user stories in Agile Projects" authors Hoa Loranger and Page Laubheimer, NN/g Nielsen Norman Group, 12 March 2017, https://www.nngroup.com/articles/ux-user-stories/ (article).
2. "Agile software development methodologies and how to apply them", author Monjurul Habib, 30th December 2013 (article).
3. "An approach for iterative and requirements-based quality assurance in DevOps" author Albert Tort, The magazine for RE professionals from IREB, Issue 2016 -03
4. A.S.Syed Fiaz, N.Asha, D.Sumathi and A.S.Syed Navaz. 2016. Data Visualization: Enhancing Big Data More Adaptable and Valuable. International Journal of Applied Engineering Research. 11(4): 2801-2804.
5. A.S.Syed Navaz, P.Jayalakshmi, N.Asha. 2015. Optimization of Real-Time Video Over 3G Wireless Networks" September. International Journal of Applied Engineering Research. 10(18): 39724-39730.
6. A.S.Syed Navaz & Dr.G.M.Kadhar Nawaz. 2016. Flow Based Layer Selection Algorithm for Data Collection in Tree Structure Wireless SensorNetworks. International Journal of Applied Engineering Research. 11(5): 3359-3363.
7. A.S.Syed Navaz and Dr. G.M. Kadhar Nawaz. 2016. Layer Orient Time Domain Density Estimation Technique Based Channel Assignment in Tree Structure Wireless Sensor Networks for Fast Data Collection. International Journal of Engineering and Technology. 8(3): 1506-1512.
8. A.S.Syed Navaz, N.Asha & D.Sumathi "Energy Efficient Consumption for Quality Based Sleep Scheduling in Wireless Sensor Networks" March - 2017, ARPN Journal of Engineering and Applied Sciences, Vol No - 12, Issue No - 5, pp.–1494-1498.
9. Boehm, Barry. "A survey of agile development methodologies." Laurie Williams (2007).
10. Carrera, Álvaro, Carlos A. Iglesias, and Mercedes Garijo. "Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development." Information Systems Frontiers 16.2 (2014): 169-182.
11. Cohn, M: User Stories Applied: For Agile Software Development. Addison Wesley (2004)
12. Fitzgerald, Brian, et al. "Scaling agile methods to regulated environments: An industry case study." Software Engineering (ICSE), 2013 35th International Conference on. IEEE, 2013.
13. FUADI, Ashar. "Introducing tcframe: A Simple and Robust Test Cases Generation Framework." Olympiads in Informatics (2015): 57.
14. Hammond, Susan, and David Umphress. "Test driven development: the state of the practice." Proceedings of the 50th Annual Southeast Regional Conference. ACM, 2012.
15. Hong, Weiyin, et al. "User acceptance of agile information systems: a model and empirical test." Journal of Management Information Systems 28.1 (2011): 235-272.
16. Hong, Weiyin, et al. "User acceptance of agile information systems: a model and empirical test." Journal of Management Information Systems 28.1 (2011): 235-272.

17. Janus, André, et al. "The 3c approach for agile quality assurance." Proceedings of the 3rd International Workshop on Emerging Trends in Software Metrics. IEEE Press, 2012.
18. Martin, R., &Melnik, G. (2008). Tests and requirements, requirements and tests: a m¨obius strip. IEEE Software, 25(1), 54–59.
19. Mugridge, R., & Cunningham, W. (2005). Fit for developing software: framework for integrated tests. Upper Saddle River, NJ:Prentice Hall.
20. N. Asha, et al. "Customer segregation in banking organisation using knowledge management." IJPT Vol. 8 Issue No.3 (Sep-2016): 17645-17649
21. Nuutila, Esko, and Eljas Soisalon-Soininen. "On finding the strongly connected components in a directed graph." Information Processing Letters 49.1 (1994): 9-14.
22. Pohl, Klaus. Requirements engineering: fundamentals, principles, and techniques. Springer Publishing Company, Incorporated, 2010.
23. Tort, Albert, Antoni Olivé, and Maria-Ribera Sancho. "An approach to test-driven development of conceptual schemas." Data & Knowledge Engineering 70.12 (2011): 1088- 1111.
24. Tort, A.: The Recovery Approach. Available at: https://re-magazine.ireb.org/issues/2015-1-ruling-complexity/the-recover-approach/
25. TestLink Open Source Test Management. Available at: https://testlink.org
26. Wynne, M.: The cucumber book: behaviour-driven development for testers and developers. Pragmatic Bookshelf (2012).