ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(20), DOI: 10.17485/ijst/2016/v9i20/91223, May 2016* ISSN (Online) : 0974-5645

Integration Testing Prior to Unit Testing: A Paradigm Shift in Object Oriented Software Testing of Agile Software Engineering

Shaik Mohammad Shahabuddin* **and Y. Prasanth**

Department of Computer Science and Engineering, K L University, Guntur - 522502, Andhra Pradesh, India;
shaik.shahabuddin@gmail.com, prasanthyalla@kluniversity.in

Abstract

Objectives: The main objective of the paper is to investigate the paradigm shift in software testing with hypothesis "integration testing should precede unit testing for cultural transformation resulting in economic governance". **Methods:** The research method employed is real time as the authors involved in the development process in two reputed software companies. We considered five projects namely Vehicle Monitoring System (VMS), Easy Subcontracting (EZSUB), Key to Engineering (K2ENGG), Electronic Material List (EMLIST), and Feedback Database (FBDB). They are developed in Agile methodology. We observed the testing time with traditional approach and also the proposed approach which puts integration testing prior to unit testing. **Findings:** Our research in association with leading software development companies revealed that integration testing prior to unit testing has significant impact on accelerating and transforming to increased agility. Thus the hypothesis is tested and proved to be positive. This new approach in testing will go a long way in achieving agility at scale with increased success rate. Our findings can be summarized as follows. Integration testing prior to unit testing can take software enterprises towards agility at scale. The new paradigm shift can cause cultural transformation from traditional approach to economic governance through agile methods. It also could improve in various parameters such as time-to deliver products, efficiency in software development and delivery process, increased personnel morale and increased client satisfaction besides achieving high success rate in the context focusing agility at scale. The proposed approach resulted in performance improvement in all five projects. The average percentage increase in productivity and delivery for VMS is 27.62%, EZSUB is 27.74%, K2ENGG is 31.29%, EMLIST is 30.83% and FBDB is 33.33%. **Application:** To our knowledge it is our first empirical study that threw light on the paradigm shift in object oriented software testing with different case studies.

Keywords: Agile Software Engineering, Integration Testing, Object Oriented Software Testing, Unit Testing

1. Introduction

Agile models which are being used by software development companies to deliver enterprise-level software with high success rate have changed the way software development orientation and delivery orientation. According to[1] integration testing should precede unit testing. This will make a paradigm shift in software development and delivery strategies and is an important area of research which will have high impact on the software industry and client community.

Agile methodology is the modern software process model that can adapt to dynamically evolving markets[2]. Moreover agile approach in software engineering has increased the expectations of the industry[3]. Capability Maturity Model Integration can be combined with agile methodology to have the synergic benefits with agile software development[4]. Gurpreet et al.[5] explored agile methodologies and realized that agile methods provide advantages like frequent delivery, customer satisfaction, transparency, flexibility, improved productivity, better software quality, and predictability. Shahla and Lars[6] investigated barriers in knowledge sharing in agile software engineering. They could identify 37 specific barriers and advocate having further research in this area.

In[7] explored to have an expert decision making

* *Author for correspondence*

system for choosing right process model in empirical engineering. In[8] explored the agile method SCRUM in the context of distributed software development to investigate its merits. The merits include the productivity and reduced work hours. Evelyn et al.[9] explored the success factors of agile models. They opined that the size of project does not influence success but the degree of adaptation and value congruence. In[10] opined that rapid application development is possible with agile software engineering. Manuel et al.[11] explored the principles of agile software engineering that include the agility in development and delivery.

1.1 More on Agility

In[12] explored how to measure agility and architectural integrity. The agile methodology brought culture changes in the industry. As measured improvement can help in monitoring and quantifying improvements, it is essential to define present capability, capability improvement, and have a roadmap to fulfil it besides monitoring progress. Strong measurement practices can improve the productivity of software development teams. In[13] explored the economics to be achieved in software development by using pair programmers. Economy can be achieved by employing a developer into two projects simultaneously. In[14] explored the possible improvements in agile methodologies for real world application development with measures. Agile scaling model was focused by Scott[15] for improved agility. The agile scaling model encompasses many aspects of the development and delivery processes. Agility at scale through agile development and agile delivery was the main focus of the research. Towards this end, agile scaling model has provided a roadmap.

Hillel et al.[16] compared agile methodology and CMMI. They advocated the possible usage of both of them in an organization in order to exploit the synergic effect of combining them. The reason for believing that these two are exclusive include misuse, lack of accurate information, terminology difficulties and top-down vs. bottom-up improvements. They concluded that both will complement each other. DCMS[17] explored project management and governance for better results. In[18] investigated industry best practices in software development. He focused on self-organizing agile teams, agile methods, agile development, and success of agile methodology. He advocated that agile methodologies provide many benefits to software industry.The relationship between agile model and CMMI and measuring agility models are focused in[19,20] and [21].

In[22] explored risk management in software development including testing. With respect to complex software systems, it is essential to know things before something happens. Towards this end, this researcher explored failure and success cases that can provide useful insights into software development with high success rate. Recently[1] reinforced the need for the transformation and paradigm shift in agile development and delivery processes. Our research in this paper focuses on the impact of the transformation from traditional approach to the new approach where in integration testing prior to unit testing by demanding intermediate mile stones contain test cases of integrated functionality. This will pave way for early delivery of software product to clients and lead to trustworthy communications among all stakeholders of the product.

1.3 Differences between Traditional and Agile Economic Practices

There are subtle differences between traditional approaches and agile approaches in software delivery and governance. The traditional software development has well defined and distinct phases while economic governance advocates continuously evolving systems. The development team and maintenance team in traditional governance have clear distinct handoff while those teams involve in common platform, process, development and maintenance in economic governance[1]. The traditional governance is based on sequential activities from analysis to testing while the economic governance focuses on set of usable capabilities that constantly add value to the client. Role specific tools and processes are involved in traditional approach while agile approach has collaborative platform with web-based tools, integration and practices. The plans in traditional approach are prone to false precision while the agile model has honest and evolving precision while it can resolve uncertainties. The traditional governance is based on measurement of activities and artefacts while the agile approach measures incremental outcomes and quality of the outcomes. The engineering discipline is used for tracking progress against static plans while the economic discipline strives to reduce uncertainties, adapt and steer changes, measure trends and manage variance effectively[1].

Our contributions in this paper are as follows. We designed and implemented a framework that promotes integration testing prior to unit testing. We investigated the benefits of this approach with industry case studies. We evaluated our approach and the results revealed that

this approach can result in agility at scale besides helping to achieve time-to-market products in software industry.

The remainder of the paper is structured as follows. Section 2 presents the methodology and proposed framework. Section 3 describes experiments and results. Section 4 evaluates our approach while section 5 concludes the paper besides giving directions for future work.

2. Research Methodology

Since agile methodology and its underlying experiments and observations need a real time environment, the authors took special interest in conducting research. This section provides the methodology followed. The authors approached 15 reputed software development companies where they have been practicing agile methods for many years. However, only two companies accepted the authors to participate in the research in their premises for one year. These companies were equally interested in working on the hypothesis aforementioned in this paper. For referring to the two companies we use the terms *COM1* and *COM2* respectively.

Both the companies were about to start projects using agile methodology. The authors requested the respective authorities in both companies for conducting research for one year along with their engineers. The objective was to explore the hypothesis "integration testing should precede unit testing for cultural transformation resulting in economic governance" as it can have significant impact on the software industry and other stakeholders. *COM1* agreed to test the hypothesis using three case studies while *COM2* agreed to carry out research with two agile projects. Though there is plethora of advantages when agile method is followed, our focus was on the said hypothesis. The authors spent 1 year time to observe the five projects while practicing integration testing prior to unit testing.

While practicing the new paradigm shift in testing agile projects i.e., integration testing prior to unit testing is carried out for increased agility and demand intermediate milestones containing executable test cases of integrated functionality of the systems. This is meant for discovering architecturally significant challenges or big uncertainties early in the SUT and increase agility in delivering product to client. Once this is done, its impact is understood in terms of project management, agility at scale, time, faster delivery, and time to market. We kept on participating and recording the milestones and results in agile paradigm. Then we evaluated the results with ground truth values of similar projects.

2.1 Proposed Architectural Framework

This section provides the methodology that helps in completing this research which aims at focusing integration testing prior to unit testing which is essentially a paradigm shift in object oriented software testing of agile software engineering. Towards achieving this aim a framework is designed and implemented that will demonstrate the proof of concept. Though the real world agile process models have many aspects, the proposed framework is confined to the theme of "integration testing prior to unit testing" and quantifying its benefits when applied in software development life cycle. The conceptual model of the methodology is presented in Figure 1.

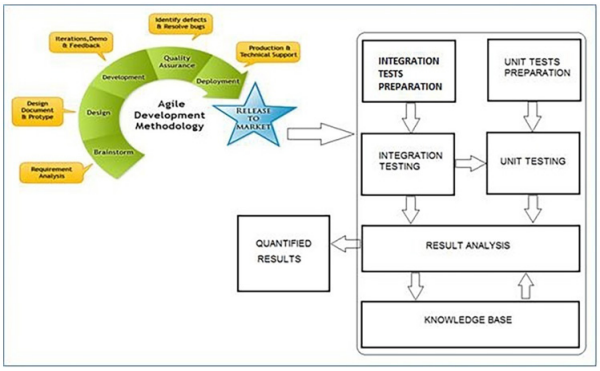

Figure 1. Conceptual model of the proposed methodology.

As shown in Figure 1 it is evident the agile development methodology results in a product that is considered as Software under Test (SUT). For the given SUT integration tests and unit tests are prepared. For SUT, the product which has been built in agile methodology and available with documentation is used. After identifying SUT, integration tests and unit tests are prepared. The integration testing is done prior to unit testing. This is to ensure that the intermediate milestones include executable test cases of integrated functionality. This will ensure that big uncertainties are resolved first. It is human nature to address the easy things first to show early progress, but there is no economic leverage in doing so. By postponing the resolution of uncertainty, you decrease the probability of success. Demanding an integration-first priority will help you increase the agility of the larger project team. The results of testing are analysed and compared with the knowledge obtained from knowledge

base. The analysis results in quantifying the results with respect to parameters that are pertaining software quality and economy of scale.

The quantified results can provide evidence that the integration testing prior unit testing will improve the agility and help software development teams to have better project management, analysis, architecture, design and development and testing. Economic governance with the proposed methodology is possible with continuously evolving systems, common process, platform, and measures.

3. Case Study: Vehicle Monitoring System (VMS)

We involved in five case studies. We provide more details of them later. However, this section focuses on the VMS which demonstrates how integration testing prior to unit testing led to realize various benefits such as faster delivery, time to market, and economy at scale. How the transformation took place from traditional agile approach to agility at scale with the new paradigm shift in testing object oriented software systems. This is very big project that may take several years. However, the functionalities shown in Figure 2 were focused in the period of 1 year. The project continues with more functionality for few more years as we estimated at the time of writing this paper.

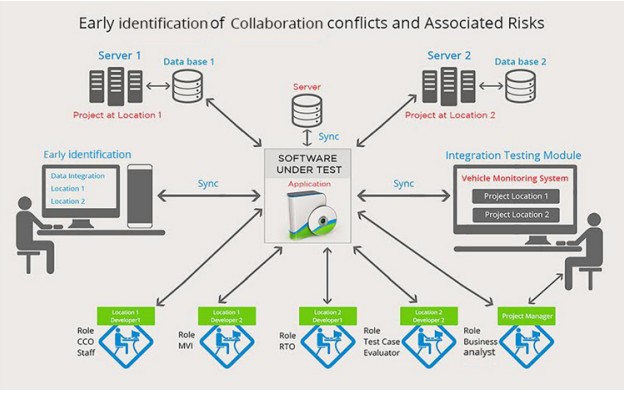

Figure 2. Shows the dynamics of the VMS case study.

The VMS project has many users in different roles such as CCO, Staff, MVI, and RTO. The application is meant for providing different dependent functionalities with respect to vehicle registration and other associated permits at government office pertaining to road transport. There are many roles in the application. Agile methodology is followed in the work. The whole project is divided into different user stories. User stories are taken up with short sprints with 15 days' time for each sprint. The main focus was to investigate the benefits of integration testing prior to unit testing.

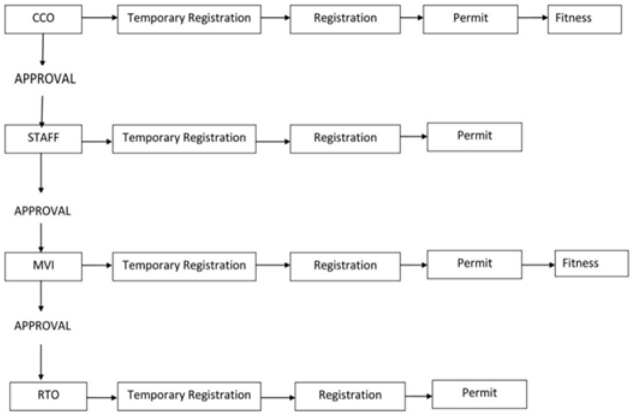

Figure 3. Different user stories and the integrated scenario of VMS.

As can be seen in Figure 3, it is evident that the user in CCO role performs temporary registration process which includes registration, permit and fitness to new vehicles. His approval is crucial for the next module which takes care of staff functionalities. The users in staff role verify everything except fitness. The approval of this user goes to the user in MVI role. The MVI user again verifies temporary registration, registration, permit and fitness and gives his approval to the final authority. The user in RTO role is authorized to provide final registration and permit to vehicle owners. The bottom line here is that each module has to be integrated with other module in order to have a complete system that works as per the functional requirements. Here comes the importance of integration testing which will speed up development and delivery process. Once integration is tested with other module, it is possible to deliver the product and then follow unit tests later as well. Therefore it is evident that integration testing prior to unit testing can have its impact on the development cycle, delivery process, customer satisfaction and other stakeholders of the project. In the process it is possible to achieve economy of scale which is inevitable to adapt agile methodology for big projects as well.

As can be seen in Figure 4, it is clear that RTO is the role in which user logged in. The application number, status and application date are shows at the top. The applicant details, vehicle details, tax details and temporary

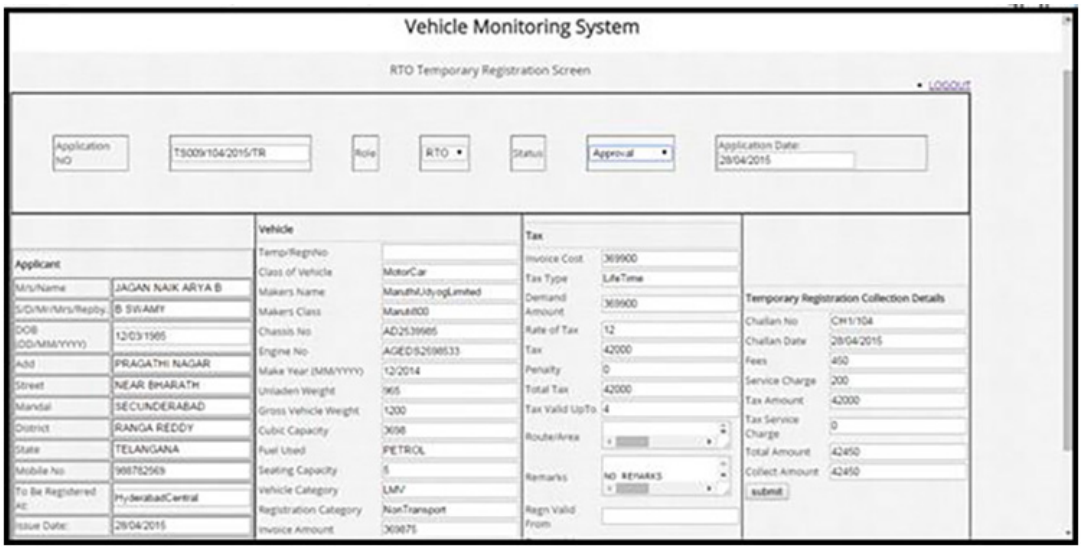

Figure 4. Shows RTO temporary registration screen.

registration collection details are presented in the UI screen. The approval of the RTO user is final in the project.

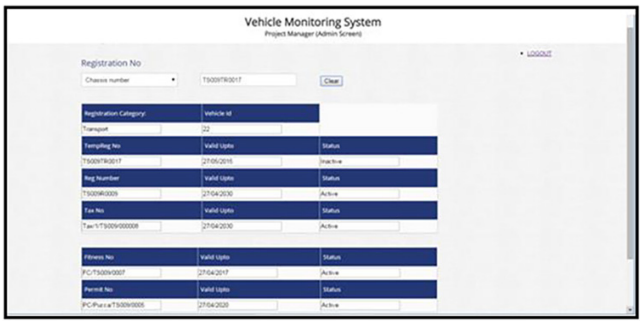

Figure 5. Shows integration testing results.

Figure 5 shows that, the UI is meant for showing status of integration testing by combining all increments or modules. The successful integration of each sprint or module or increment is reflected very clearly here. This is the screen for admin who can understand the integration status of all modules in the project. When integration testing is done prior to unit testing, it is possible that development team and delivery team can get rid of stress and plan for delivering the updated software to client. The client feedback anyway can provide more insights into the iteration and helps in updating product with ease. There are many benefits of this approach. First, it provides flexibility in development and delivery processes. Second, it promotes "time to market" feature of software. Third, it is possible to expedite development and delivery process in agile methodology. Fourth, it encourages team to give importance to integrated functionality as programs are generally tested at the time of development. Fifth, it saves time and money thus leading to economy of scale. Sixth, the benefits of integration testing prior to unit testing go a long way to impact software industry to adapt agile methodologies for bigger projects. Seventh, it helps all stakeholders including clients get benefits in terms of time and money besides time to market product deliverables.

3.1 Research Results

As described earlier the authors are involved in 5 agile projects namely Vehicle Monitoring System (VMS), Easy Subcontracting (EZSUB), Key to Engineering (K2ENGG), Electronic Material List (EMLIST), and Feedback Database (FBDB). The results or observations are presented here.

Table 1. Summary of agile projects under investigation

Project	Number of Iterations/ Sprints	Average Sprint Time	Company
VMS	10	3 Weeks (15 days)	COM1
EZSUB	14	3 Weeks	COM1
K2ENGG	15	2 Weeks	COM1
EMLIST	12	2 Weeks	COM2
FBDB	10	2 weeks	COM2

As shown in Table 1, there are five projects for which the new paradigm is employed. VMS has 10 sprints, EZSUB 14, K2ENGG 15, EMLIST 12 and FBDB 10 sprints. The table also displays the company in which

project was carried out and the number of weeks allotted for the sprints in agile methodology.

3.2 Results of VMS

This section provides the results of VMS project.

Table 2. Results of VMS

Sprint	Usual Test Time (Including integration)	Test Time with New Approach	Testing Productivity	Delivery Productivity
1	7	6	1	1
2	10	8	2	2
3	11	7	4	4
4	11	8	3	3
5	11	8	3	3
6	11	8	3	3
7	11	8	3	3
8	11	8	3	3
9	11	7	4	4
10	11	8	3	3

As shown in Table 2, the usual testing time, testing time with new approach, testing productivity and delivery productivity are provided for VMS project.

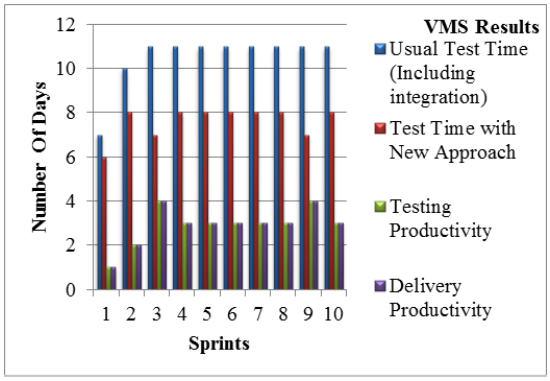

Figure 6. Productivity analysis for VMS.

As shown in Figure 6, it is evident that the productivity observed when integration testing is done prior to unit testing is presented. The horizontal axis shows the sprints in the project while the vertical axis represents the number of days. The results revealed the observations pertaining to usual test time including integration, test time with the new approach, testing productivity achieved and the delivery productivity achieved for VMS project.

3.3 Results of EZSUB

This section presents the empirical results of EZSUB project.

Table 3. Results of EZSUB

Sprint	Usual Test Time (Including integration)	Test Time with New Approach	Testing Productivity	Delivery Productivity
1	7	6	1	1
2	10	8	2	2
3	10	8	2	2
4	10	7	3	3
5	10	7	3	3
6	10	7	3	3
7	10	7	3	3
8	10	6	4	4
9	10	7	3	3
10	10	7	3	3
11	10	7	3	3
12	10	7	3	3
13	10	7	3	3
14	10	8	2	2

As shown in Table 3, the usual testing time, testing time with new approach, testing productivity and delivery productivity are provided for EZSUB project.

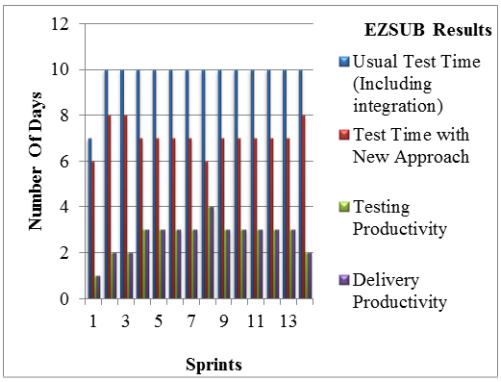

Figure 7. Productivity analysis for EZSUB project.

As shown in Figure 7, it is evident that the productivity observed when integration testing is done prior to unit testing is presented. The horizontal axis shows the sprints in the project while the vertical axis represents the number of days. The results revealed the observations pertaining to usual test time including integration, test time with the new approach, testing productivity achieved and

the delivery productivity achieved for EZSUB project. The new paradigm helped the team to achieve increased productivity and delivery performance.

3.4 Results of K2ENGG

This section presents the results of K2ENGG project.

Table 4. Results of K2ENGG

Sprint	Usual Test Time (Including integration)	Test Time with New Approach	Testing Productivity	Delivery Productivity
1	7	5	2	2
2	10	7	3	3
3	10	7	3	3
4	10	7	3	3
5	10	7	3	3
6	10	7	3	3
7	10	6	4	4
8	10	6	4	4
9	10	7	3	3
10	10	7	3	3
11	10	7	3	3
12	10	7	3	3
13	10	7	3	3
14	10	7	3	3
15	10	7	3	3

As shown in Table 4, the usual testing time, testing time with new approach, testing productivity and delivery productivity are provided for K2ENGG project.

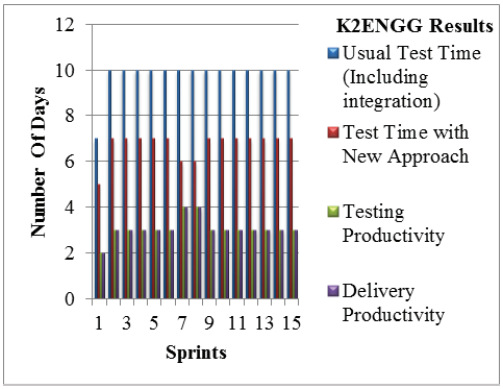

Figure 8. Productivity analysis for project K2ENGG.

As shown in Figure 8, the productivity of the team improved as there is clear decrease in the number of days taken for testing increments and deliver product to client. The testing productivity and delivery productivity are increased by 2 – 4 days in each sprint. These results were captured when the team was working for project K2ENGG.

3.5 Results of EMLIST

This section presents the empirical results of EMLIST project.

Table 5. Results of EMLIST

Sprint	Usual Test Time (Including integration)	Test Time with New Approach	Testing Productivity	Delivery Productivity
1	7	6	1	1
2	10	8	2	2
3	10	7	3	3
4	10	7	3	3
5	10	7	3	3
6	10	7	3	3
7	10	7	3	3
8	10	7	3	3
9	10	6	4	4
10	11	6	5	5
11	12	8	4	4
12	10	7	3	3

As shown in Table 5, the usual testing time, testing time with new approach, testing productivity and delivery productivity are provided for EMLIST project.

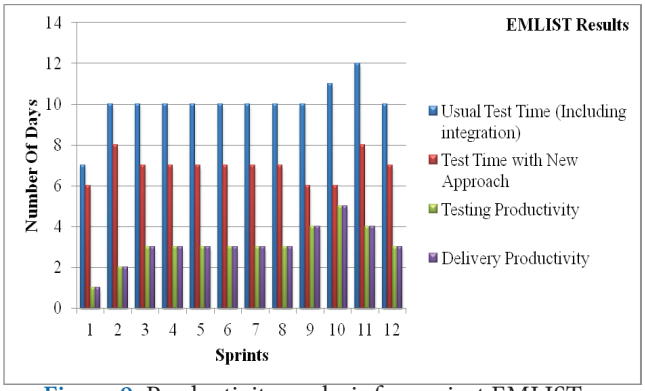

Figure 9. Productivity analysis for project EMLIST.

As shown in Figure 9, the productivity of the team improved as there is clear decrease in the number of days taken for testing increments and deliver product to client. The testing productivity and delivery productivity are increased by 1 – 5 days in each sprint. These results were

captured when the team was working for project EMLIST.

3.6 Results of FBDB

This section presents the empirical results of FBDB.

Table 6. Results of FBDB

Sprint	Usual Test Time (Including integration)	Test Time with New Approach	Testing Productivity	Delivery Productivity
1	7	6	1	1
2	10	7	3	3
3	10	7	3	3
4	10	7	3	3
5	10	6	4	4
6	10	6	4	4
7	10	7	3	3
8	10	6	4	4
9	11	7	4	4
10	11	7	4	4

As shown in Table 6, the usual testing time, testing time with new approach, testing productivity and delivery productivity are provided for FBDB project.

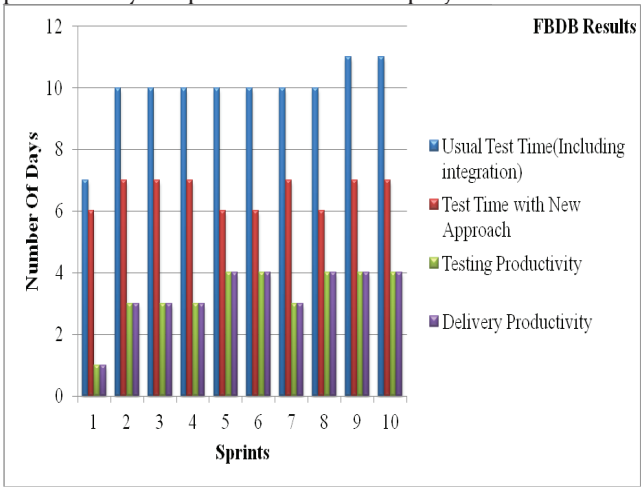

Figure 6. Productivity analysis of project FBDB.

As shown in Figure 10, the productivity of the team improved as there is clear decrease in the number of days taken for testing increments and deliver product to client. The testing productivity and delivery productivity are increased by 1 – 4 days in each sprint. These results were captured when the team was working for project FBDB.

3.7 Customer Satisfaction Dynamics

This section presents the empirical results pertaining to customer satisfaction. Especially it shows the satisfaction level of customers with new approach and old approach.

Table 7. Customer Satisfaction Index with New Approach

Project	New Approach (score out of 10)
VMS	9
EZSUB	8
K2ENGG	8
EMLIST	9
FBDB	9

As shown in Table 7, the customer satisfaction level when new approach is used is increased when compared with that of old approach presented in Table 8.

Table 8. Customer Satisfaction Index with Old Approach

Project	Old Approach (score out of 10)
Project-A	7
Project-B	6
Project-C	7
Project-D	6
Project-E	7

As shown in Table 8, the customer satisfaction level when old approach is used is less when compared with that of new approach presented in Table 7.

4. Evaluation

The proposed methodology has been implemented and evaluated on five case studies. The basis for evaluation is the hypothesis we conceived and tested with empirical study. The hypothesis is "integration testing should precede unit testing for cultural transformation resulting in economic governance". To complete our evaluation the following research questions are investigated.

- RQ1: How integration testing prior to unit testing helps in realizing agility at scale?
- RQ2: How integration testing prior to unit testing can help realize productivity?
- RQ3: How integration testing prior to unit testing can improve customer satisfaction?
- RQ4: How the paradigm shift in testing can change culture in software industry?

RQ1 concerns the usefulness of the proposed

methodology so as to help software engineers to adapt agile methodology for bigger projects. RQ2 concerns about productivity. RQ3 concerns whether the new paradigm can improve the customer satisfaction. RQ4 concerns how the paradigm shift can impact the software industry to adapt to the new culture. The proposed methodology considered only agile projects as we believed that the proposed methodology works for only that process model. The authors of the paper collaborated with two software companies in Hyderabad, India to have a real time environment and empirical study.

The five projects in two companies were selected to investigate the effects of paradigm shift. Authors spent one year time in the investigation along with respective project teams. Since the new paradigm demands intermediate milestones contain executable test cases of integrated functionality of the systems, it resulted in productivity and faster delivery of sprints and integrated incremental systems. We recorded milestones or completion of user stories, number of sprints, time taken for testing, and number of days saved and so on. Agility at scale was the expected result in the new paradigm. With new paradigm it is possible to apply agile method to big projects due to the productivity. The agile delivery teams consistently interacted with clients and recorded the level of client satisfaction. The satisfaction was found more when compared with previous projects in which the new methodology was not followed.

The agile team members were suspecting the new methodology when projects were started. However, they were convinced as time went on due to the visible advantages of following the new paradigm in testing, development, and delivery procedures. An important observation in the attitude and culture of agile team members who participated in the new paradigm is that they were thoroughly convinced about the stated advantages of the approach. The rationale behind this is that we were able to see the statistics on productivity, time saving in testing, agility in delivery and customer satisfaction. It is very clear from Table 7 and Table 8 that the Customer Satisfaction Index (CSI) has increased. The investigation into the RQs brought about insights that proved the hypothesis positively.

4.1 Threats to Validity

We investigated on the hypothesis with five case studies where agile methodology was used with the intention to move towards agility with scale. We involved in the research of five projects in two reputed software companies located in Hi-Tech city, Hyderabad, India. The five projects were big enough and we involved in them for 1 year focusing on the hypothesis "integration testing should precede unit testing for cultural transformation resulting in economic governance". The projects in which we involved are in different domains. However, our focus was only on how integration testing prior to unit testing could have its impact on cultural transformation resulted in agility at scale and economic governance. Based on the statistics we obtained in the research in terms of the timeline, cost, scale and faster delivery the conclusions were drawn in this paper. There is threat to validity to some extent when we claim that the five projects executed in two software companies in the aforementioned location are representative of all agile projects which are reasonably big in size. Evaluation procedure involves the observations in those five projects. Evaluation with more projects may have similar insights but it cannot be assumed. Another important observation with respect to threat to validity is that we spent 1 year time only we did not spend till completion of projects. However, we can say that we investigated on the relative completion of projects and the results were based on the statistics obtained from continuous deliveries of intermediate milestones.

5. Conclusions and Future Work

Enterprises in software industry are moving from agile development and delivery to large-scale agile delivery practices to achieve agility at scale. With encouraging success rate in agile methodologies, software industry has been increasing ability in development and delivery processes. Integration testing prior to unit testing is considered one of the driving forces to ensure early releases and achieve economy at scale. This is considered as hypothesis for our work in this paper. Brown et al.[1] recently reinforced the need for the transformation and paradigm shift in agile development and delivery processes. Our research in this paper focuses on the impact of the transformation from traditional approach to the new approach where in integration testing prior to unit testing by demanding intermediate mile stones contain test cases of integrated functionality. This will pave way for early delivery of software product to clients and lead to trustworthy communications among all stakeholders of

the product. "The speed of trust" as coined and believed by[23] can be achieved through increased efficiency besides reducing overheads. Our research in this paper focused on the aforementioned hypothesis. We investigated several case studies pertaining to agile methodology in association with leading software companies. Our findings can be summarized as follows. Integration testing prior to unit testing can take software enterprises towards agility at scale. The new paradigm shift can cause cultural transformation from traditional approach to economic governance through agile methods. It also could improve in various parameters such as time-to deliver products, efficiency in software development and delivery process, increased personnel morale and increased client satisfaction besides achieving high success rate in the context focusing agility at scale. Particularly the proposed approach resulted in performance improvement in all five projects. The average percentage increase in productivity and delivery for VMS is 27.62%, EZSUB is 27.74%, K2ENGG is 31.29%, EMLIST is 30.83% and FBDB is 33.33%.

Although our research has tested and proved the hypothesis "integration testing should precede unit testing for cultural transformation resulting in economic governance" to be positive, there are many directions for future work. Firstly, our work did not specifically focus on software product lines. Software product lines is a promising engineering discipline that focuses on a set of related products with commonalities and variabilities rather than a single independent product. We plan to design and implement strategies that cater to the agility at scale for software product line. Secondly, we investigate the hypothesis with distributed enterprise applications.

6. References

1. Brown AW, Ambler S, Royce W. Agility at Scale: Economic Governance, Measured Improvement, and Disciplined Delivery. Software Engineering in Practice, IEEE.2013; 873–81.
2. Barbara K, Pearl B. A systematic review of systematic review process research in software engineering.Elsevier.2013; 2049–75.
3. Shirley C, Fabio QB da Silva, Luiz Fernando C. Forty years of research on personality in software engineering: A mapping study. Elsevier.2015; 46:94–113.
4. Silva FS, Furtado SS, Lima PA, Ivanildo MDA, Vasconcelos A PLF,ZKamei FK,Lem SRD. Using CMMI together with agile software development: A systematic review.Elsevier.2015; 58:20–43.
5. Gurpreet SM, Anju M, Harmeet S, Priyanka U. Empirical Study of Agile Software Development Methodologies: A Comparative Analysis. ACM SIGSOFT Software Engineering. 2015; 40(1):1–6.
6. Shahla G, Lars M. Perceived barriers to effective knowledge sharing in agile software teams. Information Systems Journal. 2016; 26(2):91–125.
7. Claes W, Aybüke A. Towards a decision-making structure for selecting a research design in empirical software engineering. Springer Science. 2014; 20(6):1–29.
8. Ivana B, Federico C, Igor C, Elisabetta DN, Juraj F, Raffaela M. Introducing SCRUM into a Distributed Software Development Course. ECSAW.2015; 1–8.
9. Evelyn VK, Per van DW, Aske P, Joost V. An Empirical Study into Social Success Factors for Agile Software Development.IEEE.2015; 77–80.
10. Ulrike A, Barbara P. Understanding the Influence of User Participation and Involvement on System Success – a Systematic Mapping Study. Springer Science.2013; 20(1):1–54.
11. Manuel B, Hendrik M, Alexander M, Karl W. Exploring principles of user-centered agile software development: A literature review. Elsevier.2015; 21(C):163–81.
12. Walker R. Measuring Agility and Architectural Integrity. International Journal of Software and Informatics. 2011; 5(3):415–33.
13. Hakan E, Laurie W. The Economics of Software Development by Pair Programmers. The Engineering Economist. 2009; 48(4):283–319.
14. Alistair C. Agile Software Development, The Agile Software Development Series Cockburn. Highsmith Series Editor. 2001.
15. Scott WA. Agile Scaling Model: Be as Agile as You Need to Be. IBM.2011; 1–31.
16. Hillel G, Jeff D, David A, Mike K, Sandy S. CMMI or Agile: Why Not Embrace Both! Software Engineering Institute.2008.
17. DCMS. Project Management and Governance. Taking Part Survey.2012; 1–14.
18. Jones C. Software Engineering Best Practices: Lessons from Successful Projects in Top Companies. McGraw Hill.2010.
19. GandomaniTJ, NafchiMZ. Agility Assessment Model to Measure Agility Degree of Agile Software Companies. Indian Journal of Science and Technology.2014 Jan; 7(7). Doi:10.17485/ijst/2014/v7i7/47806.
20. AlshareetOM.An Empirical Study to Develop a Decision Support System (DSS) for Measuring the Impact of Quality Measurements over Agile Software Development (ASD). Indian Journal of Science and Technology.2015 Jul; 8(15). Doi:10.17485/ijst/2015/v8i15/70774.
21. GandomaniTJ, ZulzalilH. Compatibility of Agile Software Development Methods and CMMI.Indian Journal of Science and Technology.2013 Aug; 6(8).Doi:10.17485/ijst/2013/v6i8/36349.
22. Hans S. Software Risk Management A Calculated Gamble. 2006; 1–53.
23. Covey SMR. The Speed of Trust: The One Thing That Changes Everything. Free Press. 2006