Journal of Theoretical and Applied Information Technology
30th November 2017. Vol.95. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPACT OF LEAN SOFTWARE DEVELOPMENT INTO AGILE PROCESS MODEL WITH INTEGRATION TESTING PRIOR TO UNIT TESTING

[1]SHAIK MOHAMMAD SHAHABUDDIN, [2]DR.PRASANTH YALLA

[1]PhD Scholar, Department of computer science and engineering, K L University

[2]Professor, Department of computer science and engineering, K L University

E-mail: [1]shaik.shahabuddin@gmail.com, [2]prasanthyalla@kluniversity.in

ABSTRACT

The current academic thinking on integration testing prior to unit testing using agile methodology shows that it is an innovative approach little understood and practiced formally. However, this approach according to Brown et al. contributes to economic governance, disciplined delivery and measure improvement for achieving agility at scale in software industry. This has motivated us to investigate and propose a conceptual model and make an empirical study in our previous work. In this paper, we reinforce the study with a case study based approach and quantify the real benefits of the new cultural shift in testing known as integration testing prior to unit testing. In addition to this, we studied the lean software development in terms of testing and integrated it with the phenomenon of integration testing prior to unit testing. We identified many aspects of lean principles. Nevertheless, we found mind mapping and identification of infeasible test cases are two important aspects. They are associated with lean principle like removal of waste to improve productivity further in agile and lean software development environment. The empirical results revealed that productivity is increased with the paradigm shift in testing arena. The quantification of benefits in terms of productivity shows significant performance differences between traditional approach and the leagile (new term referring to lean and agile) approach in software testing.

Keywords: *Lean Software Development, Agile Process Model, Software Testing, Integration Testing Prior To Unit Testing*

1. INTRODUCTION

Right from the year 2001 in which Agile Manifesto emerged, the agile process model is one of the software process models widely used by software industry. Due to its features that contain the advantages of iterative and incremental models, it is used to have improved customer satisfaction. On the other hand, lean software development is the process of applying lean principles to software development. Lean is understood as an extension to agile process model. Lean focuses on removal of wastage. Wang et al. [1] coined the name "Leagile" for the software development that is the combination of agile and lean. Since agile community started looking at lean and its applications to software development with agile process model, the spreading of lean adaption started. Initially lean software development was looked as another agile model. However, it has got significance later and now it is recognized as a method suitable for agile process models. It does mean that lean is on top of agile and both go hand in hand. It is the application of lean principles in agile methodology. Middleton and Joyce [2] opined that lean thinking could reduce error rates to one per million units. They also said that lean has capability to double productivity in software development and other industries.

Lean principles help in software testing as well in the context of agile. Iberle [3] applied lean science to software test labs. This researcher explored lean science, productivity in lean, modularity in lean, Map for tracking work progress, visibility of progress and so on. It is understood that some test cases may become irrelevant as the software process is dynamic in nature. Time and resource wastage can lead to failure of projects, risk

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

prone and time to market situation cannot be guaranteed. In this context, this paper focuses on a methodology that is used for lean software testing on top of agile methodology. In our prior works, we focused on a novel approach in testing known as "Integration testing prior to unit testing" considering it as a paradigm shift in software testing process. Our study on this proved that the proposition "integration testing prior to unit testing can result in agility at scale, economic performance and improved customer satisfaction" is true. In this paper, our focus is to incorporate lean principles into agile methodology and make potential observations.

The remainder of the paper is structured as follows. Section 2 reviews the current academic thinking on integration testing prior to unit test, agile methodologies and the lean software development. Section 3 focuses on understanding lean science. Section 4 presents the proposed methodology. Section 5 provides the case study and results of proposed methodology. Section 6 discusses on the benefits of the proposed methodology. Section 7 concludes the paper and provides recommendations for future work.

2. RELATED WORKS

This section throws light into the review of literature on lean in agile software development. Antinyan *et al*. [4] explores risks involved in the software development when in the presence of agile and lean development. In such environment, these researchers proposed a method to identify risks involved in software code. They followed action research methodology with two big projects. They found that complexity and revision history of a source file could reveal risk areas. Wang *et al*. [1] explored application of lean in agile software development process. They found importance of lean principle in software development. They include eliminating waste, building quality in creating knowledge, deferring commitment, delivering faster, respecting the people and optimizing the whole. Middleton and Joyce [2] provided a case study to explain lean ideas in software management. The case study is related to British Broadcasting Corporation (BBC). They found three important benefits of lean software. They include quantification of software development process, simplifying management of operations, lowering risk and increasing profits.

Rodriguez *et al*. [5] made empirical study and found that lean can be combined with agile methodology. They found that lean provides less wastage kind of culture in development and delivery processes. Ahmad *et al*. [6] presented one of the lean tools known as Kanban and its usage in software development process. The researchers found that the Kanban usage improved customer satisfaction, improve quality of software development and delivery approaches. It also could improve developer motivation and communication among all stakeholders. Chuanga *et al*. [7] assessed agile software process usage and contributions in institutions and by scholars. They found that scholarly publications on agile models have increased significantly. It is observed, that agile methods usage is increased gradually. Rodriguez *et al*. [8] studied the lean thinking in different industries including software development. They focused on telecom industry with respect to lean software development in terms of strengths and challenges. They found many advantages and the challenges they found include creating lean culture, transparency, and achieving flow.

Silva *et al*. [9] made a review of the benefits of combining CMMI and agile software models. They found that using agile models in software development could help them to improve processes to level 5 of CMMI. Thus, CMMI and agile models have certain relationship. Anslow and Maurer [10] studied the teaching of agile project development as a course in education. They said that well-defined scope is important to students to do projects successfully using agile methods. Alia *et al*. [11] investigated the need for value stream mapping for process improvement in software development. Especially their study focused on large-scale software development. They found the utility of flow-assisted value stream mapping for effectiveness. Dwyer [12] provided the significance of lean and agile research. Rahman *et al*. [13] studied agile methods like XP and Scrum and found that they are very useful for continuous software deployment. Ali *et al*. [14] made a simulation study on value stream mapping in agile software development. They also found the utility of lean in agile software development process. Two industrial cases provided utility of the value stream mapping and its usage in agile methods.

Suomalainen *et al*. [15] focused on the study of continuous planning and its benefits in agile and lean software development. They found that some organizations do not follow continuous planning. The elements involved in continuous planning include organizational planning and strategic

Journal of Theoretical and Applied Information Technology
30th November 2017. Vol.95. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

planning. They found continuous planning of three lean and agile cases in terms of day, iteration, release, product, portfolio, and strategy. Fagerholm *et al*. [16] investigated on continuous team performance in lean and agile environments. They found the need for different factors like team identity, values, team spirit and communication in lean and agile methods. They said that it is important for the team to have performance awareness, interpretation of performance and performance adaptation. Yang *et al*. [17] worked on combining software architecture and agile development process to understand the mapping between them. They found the lack of knowhow in the industry with respect to combination. Ruhi and Akhigbe [18] studied lean usage in design science research and proposed a conceptual framework for lean integration. Dreesen *et al*. [19] investigated on Agile Global Outsourced Software Development (AGOSD) that involves software development using agile methods and lean principles. Kuhrmann and Munch [20] tried to find group dynamics in teaching agile methods and with respect to project management courses. They found different aspects in teamwork and found that performance depends on quality of teams with agile methods.

Osadchyy and Webber [21] studied on agile methodology for continuous and iterative development and delivery process. With agile and lean principles, they found that developers could have good communication and overcome any issues with communication. Iberle [3] applied principles of lean to manage software-testing lab. They found that lean science could improve productivity dramatically. Isomursu *et al*. [22] studied the role of user experiences in the agile models. They found that user experience was not considered agile. However, they found the utility of user experience design into agile process models for better lean transformation in organizations. They found that lean thinking in software development testing could have dramatic impact on the productivity. Kasoju *et al*. [23] did their research on Evidence Based Software Engineering (EBSE) with respect to automotive testing process. They found that EBSE is very important with respect to agile and lean practices and it can help in technology transformations to be more productive. In this paper, we investigated on the influence of lean in software testing process in the context of integration testing prior to unit testing. This kind of research, to our knowledge, is novel in nature and we found the utility of it by quantifying productivity in testing.

3. UNDERSTAND LEAN SCIENCE

It is crucial to understand lean science or lean thinking before adapting it to agile methodology. Lean refers to a set of management practices that lead to reduction of waste. It was originally developed for manufacturing industries. However, it can be adapted to software testing process as well as it involves technicalities and people orientation. When waste is reduced, the testing process is optimized and thus agility is improved further besides economy of scale. It is important to understand how the flow of work is in the organization with respect to testing. Towards this end it is essential to have a systems view as illustrated in the ensuing section.

3.1. Systems View

The key to lean science in this paper is to have a systems view, which reflects workflow through the testing process in agile methodology. Workflow refers to chunks of work moving in the system from one step to another. Sometimes, the work is in waiting state.

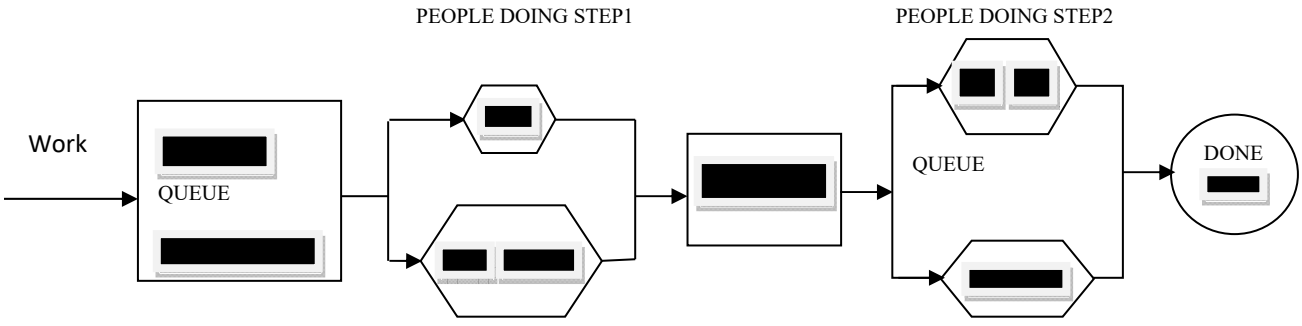


Figure 1: Shows Systems View That Can Help In Lean Implementation

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The work is divided into number of batches. Each batch has a set of chunks of work. There are two states in the system. They are work being done and waiting state. The chunks under waiting are assumed to be in a queue. The main goal of lean science is to make the batches to reach the "Done" state faster. This convergence is expected when lean is practiced. This is general systems view that can be adapted to Lean Science implementation in agile process model.

3.2. Modelling Lean

It is possible that the work pertaining to software testing is divided into number of batches. Then the progress of the work is visualized and tracked with acceptable accuracy. Then it is possible to identify unnecessary work and poorly defined and duplicate actions with ease. While modelling lean, the work is considered in accordance with the framework shown in Figure 1.

3.3. Split Testing Work into Batches

It is actually done easily in manufacturing industries. However, in this paper we are applying lean to software testing process. It is challenging as the batches are made as part of lean science with the intention of reducing waste, adding value to the process besides showing significant impact on the agility at scale. Applying lean science should increase the output of batches. Some typical examples of batches are as follows.

- Finding and writing appropriate test cases that can adequately cover a newly incorporated features in the software system, then running the system, apply test cases, discover bugs and report them properly.
- Designing and executing in-depth tests pertaining to performance of the system and reporting what went wrong in performance.
- Writing and running tests that find bugs pertaining to localization or internationalization and report defects discovered.
- Generating certain test cases automatically and executes them as a test suite for discovering bugs and reporting them.
- Designing and executing a set of regression tests to discover latent defects that are related to a new feature incorporated.
- Preparing a status report on the testing process being carried out and predicting when software can be finally released.

These batches are designed to target delivering certain value that is identified by the testing team. The list of unfixed bugs can help managers to take well-informed decisions. Effective bug triage is possible when the lean is applied as it results in discovery of bugs in a more focused way. Therefore, it is essential to divide testing work into batches to deliver value to the testing process. Batches also can reduce overhead in the testing process. Rather than providing some benefit immediately to outsiders or customers, lean science can help in reducing waste and optimize the testing process so that it results in reduction of overhead in the laboratory. When batches are determined, it is good to identify value of each batch and the exit criteria that helps in quitting work at right time.

3.4. Batch Optimization

Batch optimization can be done in different ways. This optimization can help reduce waste and improve efficiency. The optimization can be achieved by following activities such as making a cadence, limiting number of batches, and reducing batch size.

3.5. Cadence

Manage the batches on a cadence. Cadence is described here. It is a predictable activity in software testing process. It is something related to habit and habit is repeatable action. For instance, a sprint meeting takes place in the agile development environment at a fixed time every day. This kind of cadence brings about rhythm in work environment. Other cadence activities include taking backup at a regular interval (it can be automated to reduce time and effort thus reducing waste), update of web site in a time specified for maintenance (at that time visits are least to the web site). Thus, a cadence provides the following advantages.

- It saves time by reducing waste.
- It also reduces overhead and results in reduction of waste.
- When any activity is made habitually at given time, people plan to work effectively and take the advantage of cadence.
- When batches are in cadence, it is possible to prioritize and execute.
- Things done at regular intervals can be configured only once and save time besides reducing waste. For instance, booking a meeting room on every specific day of week need not be done every week. It can be done once and that is the advantage of cadence.
- Cadence can also avoid regular prioritization thus discussions on priorities is avoided which saves time and effort.
- Regularly scheduled review of work progress can help senior managers to focus more in the given time avoid disturbing teams when they are in work.

Journal of Theoretical and Applied Information Technology
30th November 2017. Vol.95. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.6. Limiting Number of Batches in Progress

It is said that limited number of batches in progress can help agile testing teams to focus more on tier work. If not, it can have negative impact on the progress of work. More number of batches in progress can have the following disadvantages.

- Increased maintenance
- Wastage of time
- Task-switching can result in waste
- Keeping track of more batches causes overhead

3.7. Reducing Batch Size

When batch size is less, it can optimize process. If not, it affects in responsiveness. Small batches provide the following benefits.

- Early delivery of results
- Faster evaluation and feedback
- Opportunities to include changes
- Reduction of risk

3.8. Visualization of Progress

When progress is visible, it is possible to keep track of progress of batches and take appropriate decisions on time. Visualization also can help discover unnecessary steps thus adding value to the process besides reducing waste. There are many visualization methods. They include Visual Stream Map, Visual Planning Board, and Cumulative Flow Diagram.

3.9. Sample Visual Stream Map (shows visible progress)

Visual stream maps provide progress visibility. Figure 2 is a sample showing the progress of a software project.

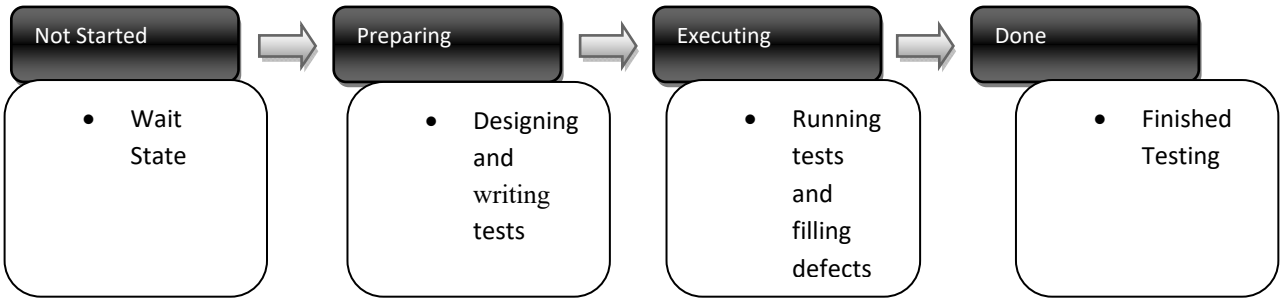
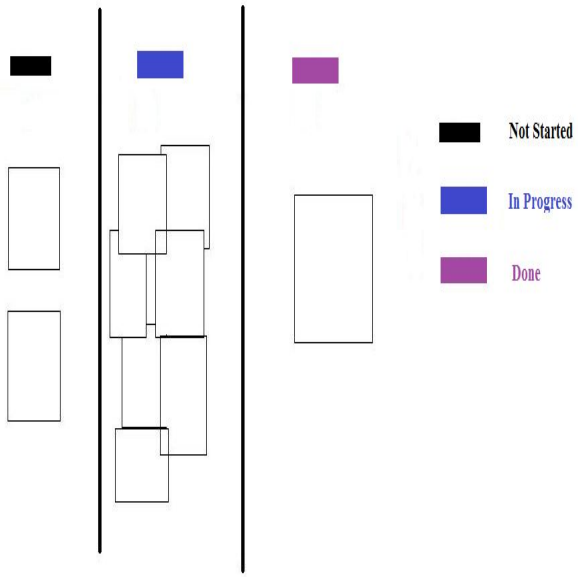


Figure 2: Shows Visual Stream Map

Visual stream map helps in understanding the progress of work just by a glance. It is useful to have control the work being carried out in the agile methodology.

3.10. Sample Visual Planning Board (snapshot in time)

At any given point of time, it is possible to monitor the progress of any software project or product. Sample visual planning board can provide good picture of the work progress. There are some tasks which are not started. Some tasks are in progress while other tasks are already completed.



Journal of Theoretical and Applied Information Technology
30th November 2017. Vol.95. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 3: Shows visual planning board

The visual planning board can help the stakeholders to have good project management activities that lead to success of software development. It improves visibility and improves customer satisfaction.

3.11. Cumulative Flow Diagram (shows progress trends over a period of time)

Cumulative flow diagram is also part of lean science in which the progress is shown clearly on weekly basis. Every week, it shows how many tasks are in progress, how many are not yet started and how many are already done. The purpose of the flow diagram is to show progress trends over a period of time.

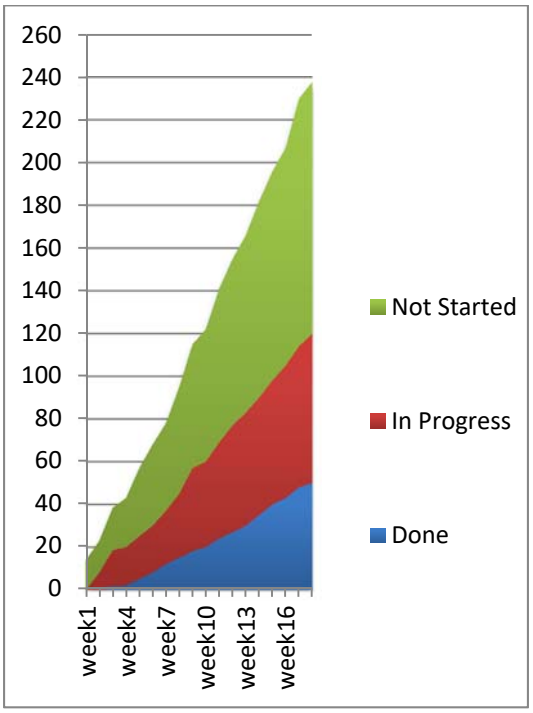


Figure 4: Shows Cumulative Flow Diagram

As shown in Figure 4, it is evident that the cumulative flow diagram shows the status of any software project. The trends in the development process are shown in every week with three data series plotted in a graph. The result reveals the portion of work which has been done, in progress or not yet started.

4. PROPOSED METHODOLOGY

This paper focuses on the impact of Lean Thinking and lean software development into Object oriented software testing with new paradigm shift in testing: Integration testing prior to unit testing. It is based on the hypothesis that is "Lean thinking into software development and testing can eliminate waste, improve quality, help deliver faster, respects people and optimizes the whole process well". This methodology has reference with our previous work, which proposed a framework to facilitate integration testing prior to unit testing, a paradigm shift in software testing.

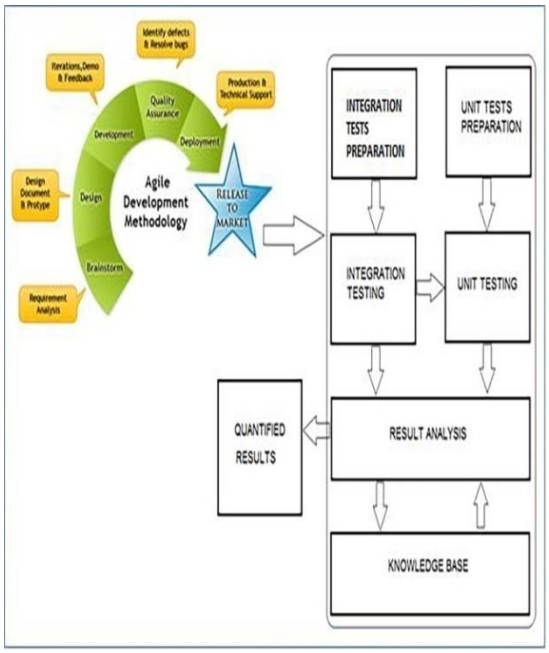


Figure 5 – Framework Facilitating Integration Testing Prior To Unit Testing [24]

As shown in Figure 5, the agile process model is used in modern software development to optimize the development process. The agility in terms of testing and the advantages of integration testing prior to unit testing were realized in our previous paper. The methodology in this paper is to focus on the lean thinking or lean science incorporated into agile methodology in the confines of the proposed framework shown in Figure 5. It throws light into the application of lean science to the software testing process. The intent of lean science is to optimize an organization or the process used in an organization. In this paper, our focus is to optimize software testing process, which

Journal of Theoretical and Applied Information Technology
30th November 2017. Vol.95. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

is based on the integration testing prior to unit testing approach proposed by us.

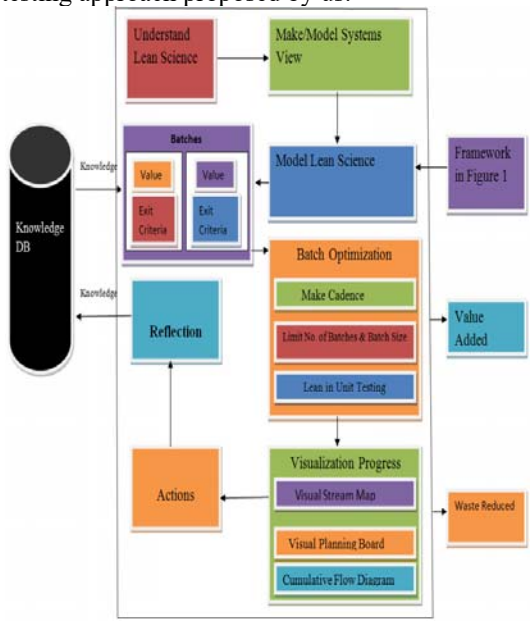


Figure 6 – Proposed Methodology To Incorporate Lean Software Development Into Agile Methodology

As mentioned earlier, this methodology has to be understood in the context of the refined agile methodology shown in Figure 6. The aim of this methodology is to know the impact of lean software development in the agile methodology when integration testing is made prior to unit testing. The methodology includes various phases. They include understanding lean science, model systems view, and model lean science, divide work into batches, optimize batches, visualize progress and take necessary actions when there is deviation from expected progress and reflection.

4.1. Empirical Approach Pertaining to Lean in Unit Testing

This section provides most important empirical study that demonstrates the proof of concept pertaining to investigating impact of lean software development into agile process model with integration testing prior to unit testing. With respect to agile software process model our prior research focused on the integration testing prior to unit testing as a paradigm shift for achieving economy of scale and delivery of agile projects on time with high customer satisfaction. This section provides validation of the approach with reiteration of that and study of impact of lean in agile process model. The investigation of lean into agile software development provided the insights presented in the previous sections.

Now the focus is on the actual empirical approach, which contributes in quantifying the performance improvement of two aspects of this research.

1. Advantages of integration testing prior to unit testing.
2. Impact of lean software development into agile process model.

After careful investigation into these two aspects, we found that the conceptual framework presented in Figure 6 is true. However, the lean parameters that can be empirically demonstrated with quantification of results are identified as follows.

a) Mind map to avoid writing unnecessary test cases

b) Identification and elimination of infeasible test cases

4.2. Elimination of Infeasible Test Cases

A coverage goal is infeasible if there exists no test that would exercise it. For some simple cases, there could be techniques that are able to identify infeasible targets; for example, dead-code detection might reveal some infeasible branches, such as the one listed in Listing 1.

```
1   public class Stack
2   {
3       int[] values=new int[3];
4       int size=0;
5   void push(int  x)
6   {
7       if(size>=values.length)
8           resize();
9       if(size<values.length)
10      values[size++]=x;
11  }
12  int  pop()
13  {
14      if(size>0)
15        return values[size--];
16      Else
17  thrownew
18  EmptyStackException()
19  }
20  private void resize()
21  {
22      int[] tmp=new int
23  [values.length*2];
    for(int i=0; i<values.length; i++)
24          tmp[i]=values[i];
25      values=tmp;
26  }
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Listing 1: Stack Implementation with Infeasible Branch [25]

As shown in Listing 1, it is evident that the test case with infeasible branch can be avoided while writing unit test cases. This can save time in writing test case time and execution time of the same. The time thus saved can be quantified and translated to productivity, which can be compared with the productivity of the approach where lean is not adapted in the form of eliminating infeasible test cases.

4.3. Mind Map for Reducing Test Cases

A fundamental goal for lean is fast-flexible flow. That is, it is useful to think of the development process as a pipeline where production takes place. Anything that slows down the pipeline causes waste. This waste includes any test case written without relevance in the process of integration testing. Mind map can reduce such test cases as this tool can provide accurate mapping to only mandatory test cases. Mind map is the graphical representation of main concepts to be tested mapped to essential test cases. Mind map shows visual representation of test cases to be written and tested. This can avoid writing unnecessary test cases. This feature is part of lean. The test cases written with mind map and without having mind map practice can differ. Therefore, this feature is considered for empirical study.

5. CASE STUDY AND EXPERIMENTAL RESULTS

A case study is considered for implementation of lean in agile besides using the integration testing before unit testing paradigm. The project is split into number of sprints. They are implemented with and without proposed approach. The experimental results are presented in terms of productivity in time for integration testing prior to unit testing and lean integration with agile model. The IT before UT was investigated in our previous work [24]. The focus of this paper is Lean Integration in terms of reducing wastage by considering infeasible branches and mind maps in testing phase. Before presenting the results of this paper, the representative results of IT before UT are presented in Table 1.

Table1: 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

The delivery and testing productivity for all the sprints is presented to quantify the utility of IT before UT. By focusing on architectural defects thru integration testing prior to unit testing could achieve the aforementioned productivity. The remainder of this section throw light into the observations of Lean Integration into agile model on top of the new paradigm shift known as IT before UT.

Table 2: Shows Unit Testing Planning, Development And Execution Details

Sprint	No. of Unit Test Cases	Planning Time (hh:mm)	Development Time (hh:mm)	Execution Time (hh:mm)	Total Time (hh:mm)
1	10	02:50	05:50	00:50	08.50
2	10	02:50	07:00	00:50	10.00
3	10	02:50	06:00	00:50	09.00
4	10	02:50	05:00	00:50	08.00
5	10	02:50	05:00	00:50	08.00
6	10	02:50	07:00	00:50	10.00
7	10	02:50	08:00	00:50	11.00
8	10	02:50	06:00	00:50	09.00
9	10	02:50	05:00	00:50	08.00
10	10	02:50	06:00	00:50	09.00

As shown in Table 2, it is understood that every sprint in agile methodology can have lean principles adapted. We found that mind mapping and identification of infeasible branches in the source code are two important aspects associated with lean principles. These two are effective in reducing wastage.

Journal of Theoretical and Applied Information Technology
30th November 2017. Vol.95. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

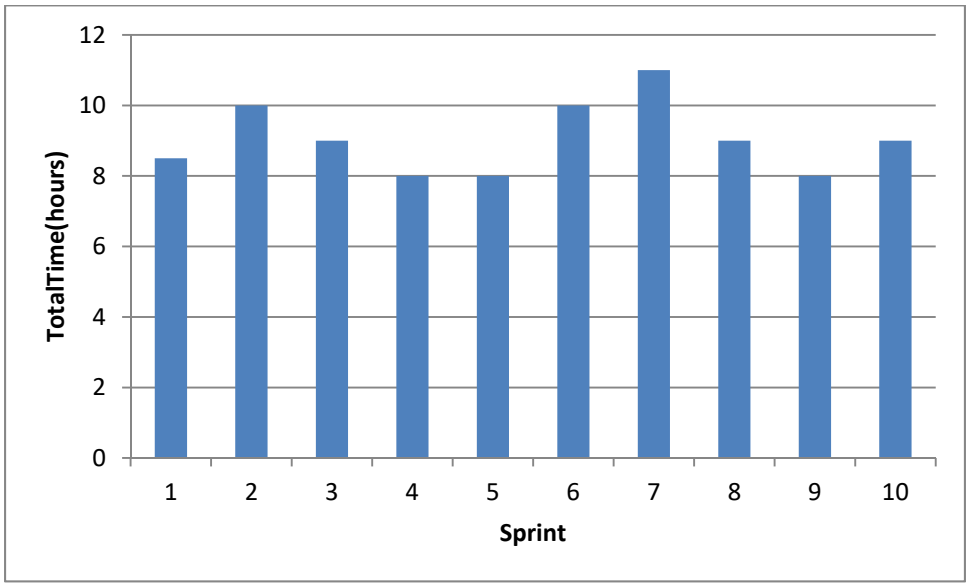


Figure 7: Sprint wise total time taken

As shown in Figure 7, it is evident that each sprint the total time taken for completing unit test cases with reference to Table 2, the number of sprints and the corresponding total time of completion of test cases for each sprint respectively.

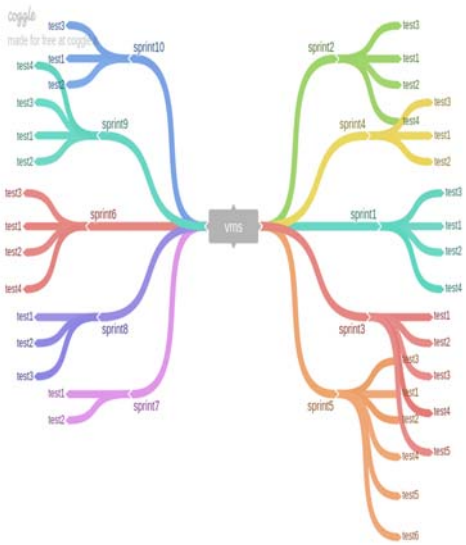


Figure 8: Shows the mind map used to adapt lean testing in agile process of VMS project

As shown in Figure 8, it is evident that the VMS project is divided into 10 sprints. Each sprint has different number of test cases. This mind map can help test engineers to quickly complete the task of writing test cases, test data generation and execution of test cases. It also helped in removal of unnecessary test cases. The effect of mind maps observed empirically is reflected in Table 3. Mind map is drawn using online software known as Coggle software [26].

Table 3: Shows Expected Outcomes

Sprint	Time Saved (Infeasible property)	Time Saved(Mind Map property)	Testing Productivity (in Man Hours)	Dollars Saved($18 per hour)
1	3	4	7	126
2	2	4	6	108
3	2	7	9	162
4	3	6	9	162
5	3	6	9	162
6	2	5	7	126
7	1	6	7	126
8	1	5	6	108
9	2	7	9	162
10	1	5	6	108

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

As shown in Table 3, the results of empirical study are presented. The number of test cases reduced due to the adaptation of the lean principles is shown. Since lean eliminate wastage in software testing process, the utility of the mind maps and identification of infeasible branches in source code are presented.

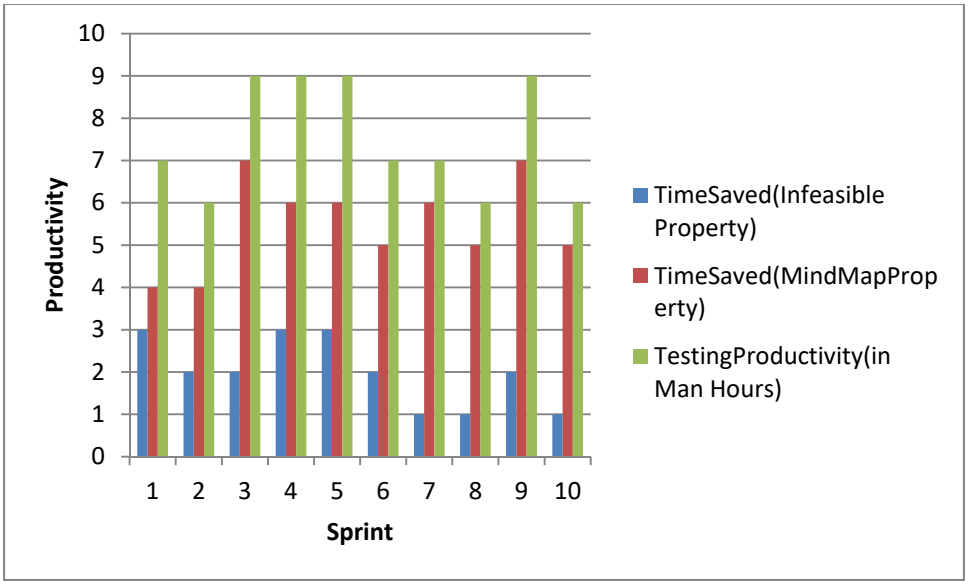


Figure 9: Shows Outcomes of Proposed Approach

As presented in Figure 9, it is understood that the productivity is increased in each sprint. The productivity is indirectly visible in the form of the number of test cases reduced in each sprint due to lean principles used in testing. Particularly usage of mind map and identification of infeasible test cases could reduce the number of test cases needed. It does mean that it was able to avoid wastage by not writing unnecessary test cases in the software testing where lean in adapted on top of agile methodology.

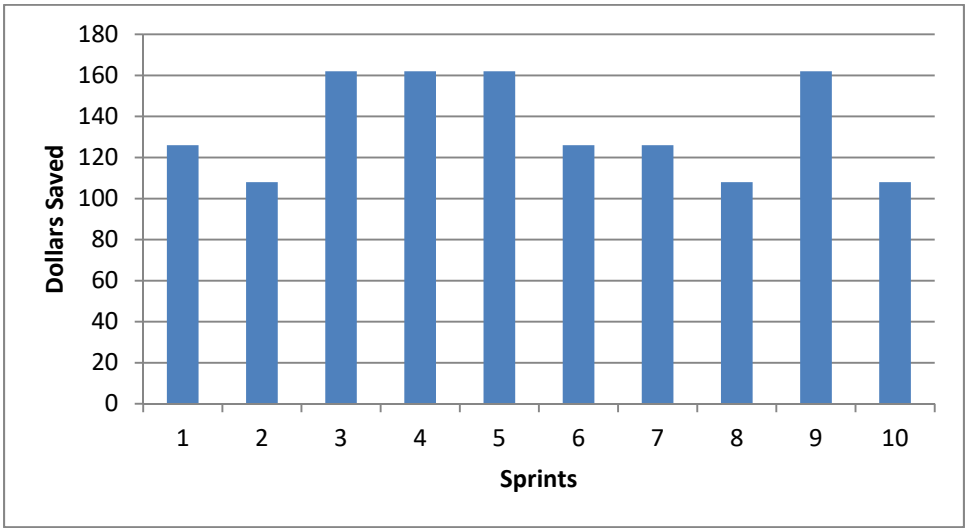


Figure 10: Shows productivity in terms of dollars saved in each sprint

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

As shown in Figure 10, there are 10 sprints and the details of money saved ($) due to the lean approach which gets rid of wastage in terms of removing infeasible branches in testing and usage of mind maps. This productivity is in agile model where lean is enforced on top of IT before UT paradigm. The results reveal that testing phase alone is capable of increasing efficiency by saving dollars. Saving $18 per hour is significant performance improvement and a total of $1350 for all sprints emphasizes the proposition conveyed in the results.

6. DISCUSSION

The combination of agile methodology and lean software development yielded synergic benefits. This section throws light into the benefits bestowed by such development environment coupled with integration testing prior to unit testing paradigm. Integration testing prior to unit testing is to achieve transformation from conventional engineering and governance to more agile economic governance as envisaged in [27]. Another important aspect of paradigm shift is to have control over agility by exploiting lean in software development and testing. Our research in this paper focused on testing. In fact, it investigated three aspects in software engineering. They are transition from traditional testing to integration testing prior to unit testing with agile, lean testing incorporated, benefits to all stakeholders when IT before UT is coupled with lean. IT before UT, as shown in our previous work [24], ensured productivity. The lean implementation in the form of mind maps and infeasible test cases (avoiding wastage) resulted in testing productivity as shown in Table 3.

The stakeholders associated with a project done in agile with lean implementation are development team, product owner, client, QA Team, Scrum Master (management) and users. The main advantage to software development team is that their mindset is changed from development orientation to delivery orientation. In addition to this, they could save 15 to 20% time and effort on reworking of code as opposed to 40% observed with traditional approach [28]. It is achieved with the proposed methodology as IT before UT, mind maps and infeasible test cases could eliminate wastage and improve testing productivity by 25% in each sprint. This is very significant leap forward in the economic governance in software engineering. By discovering architecturally significant challenges as explored in [27], IT before UT could resolve big uncertainties earlier and made inroads to faster deliveries of sprints. Thus, the aforementioned benefits are realized by development team. The side effects of these benefits include reduction of stress among team members, customer satisfaction due to speedy deliveries to client instead of spending more time on unit testing. This could improve the confidence of team members to get the culture of delivery orientation. An analogy from [27] which is, 10% reduction in complexity → 10% process improvement → 10% more capable team → 10% increase in automation, closely fits here

Economic governance, engineering governance and delivery orientation are improved with agile model where IT before UT and lean are applied. Benefits to client is that, client is able to see the product (after each sprint) early and happy to execute and give feedback. Client involvement in the development process is the key in success of the project. More details on the quantification of client satisfaction with IT before UT can be found in our previous work [24] as shown in Table 1. Measured improvement is made possible with agile, IT before UT and lean. Management of software Development Company gained more evidence and confidence in the delivery focused environment. Customer satisfaction boosted their morale and it paved way for stronger relationship with client. The management related to problem area are benefited with time-to-market products that leverage technology adaptation, change management and automation. When architectural inconsistencies are addressed first with IT before UT, it resulted in timely deliveries to client. Pressure on all stakeholders is reduced. Continuous evolvement, value-adding approaches, honesty in dealing with uncertainties, governance through proper measurements, and control over agility are the success factors observed in this research. To sum it up "less overhead and more freedom is realized by practitioners while stakeholders achieved predictable productivity and better measurement".

Based on the efforts and time saved, cost estimation is made with COCOMO II model [29]. It is observed that with agile, lean and IT before UT, the cost of production is reduced by 25 to 30% for the software projects studied in this paper, especially VMS. Cost of software engineering is significantly reduced with the proposed methodology. Project management is also affected by this positively. However, there were some challenges encountered by development team. Most of the challenges were related to change management as the IT before UT is new to the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

team. For instance, they are accustomed to write unit test cases and avoiding them and directly writing integration test cases became a problem. To overcome these challenges, a training program was conducted to the team for having better understanding and co-operate with the change brought into the system. With respect to mind maps, it is understood that they became good documentation for the stakeholders. As discussed in [30], the mind maps provide predictable outcomes besides guiding project management. They could improve information recall as they are converted to technical documentation. Potential opportunities and risks could be identified with mind maps incorporated as part of lean in agile software development.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we studied lean software testing as part of lean development and its integration with agile methodology. This study is made in the context of the paradigm shift in testing arena known as integration testing prior to unit testing. As formal unit testing consumes more time and effort, it can be avoided or minimized by following the novel approach integration testing prior to unit testing. This can help agile teams to realize more agility in delivering project incrementally to client and gain feedback instead of delaying the delivery. This strategy is said to have significant impact on customer satisfaction and revenues of software companies. However, it is little understood and practiced by agile teams. We threw light on this by investigating on the new phenomenon integration testing prior to unit testing with multiple case studies in our previous work. We found the utility of this approach in the context of agile methodology. In this paper, we reinforced our study with lean integration. The lean principles such as finding infeasible branches and avoiding test cases for that and having mind maps prior to developing test cases are employed with agile methodology. We used VMS case study to explore the agile methodology with 10 sprints and for each sprint, we quantified the productivity of integration testing prior to unit testing approach and lean software testing approach. The empirical results revealed significant performance improvement in terms of productivity and frequent delivery of product increments to customers. In future, it interesting to build formal metrics for finding benefits of lean and the new paradigm shift that is integration testing prior to unit testing.

REFERENCES

[1] Xiaofeng Wang, Kieran Conboy and Oisin Cawley, “Leagile software development: An experience report analysis of the application of lean approaches in agile software development”, *Jss*, 2012, p1-13.

[2] Peter Middleton and David Joyce, “Lean Software Management: BBC Worldwide Case Study”, *IEEE*, Vol. 59, No.1, 2012, p1-13.

[3] Kathy Iberle, “Lean in the Software Test Lab”, *Excerpt from PNSQC*, 2013, p1-13.

[4] Vard Antinyan, Miroslaw Staron, Wilhelm Meding, Per Österström, Erik Wikström, Johan Wranker, Anders Henriksson, Jörgen Hansson,Ericsson AB and AB Volvo, “Identifying Risky Areas of Software Code in Agile/Lean Software Development: An Industrial Experience Report”, *IEEE*, 2014, p1-10.

[5] Pilar Rodríguez, Jari Partanen, Pasi Kuvaja and Markku Oivo, “Combining Lean Thinking and Agile Methods for Software Development A Case Study of a Finnish Provider of Wireless Embedded Systems”, *IEEE*, 2014, p1-10.

[6] Muhammad Ovais Ahmad, Jouni Markkula and Markku Ovio, “Kanban in software development: A systematic literature review”, *IEEE*, 2013, p1-8.

[7] Sun-Wen Chuanga, Tainyi Luora and Hsi-Peng Lu, “Assessment of institutions, scholars, and contributions on agile software development”, *Jss*, 2014, p1-18.

[8] Pilar Rodríguez, Kirsi Mikkonen, Pasi Kuvaja, Markku Oivo and Juan Garbajosa, “Building Lean Thinking in a Telecom Software Development Organization: Strengths and Challenges”, *ACM* , 2013, p1-10.

[9] Fernando Selleri Silva, Felipe Santana Furtado Soares, Angela Lima Peres, Ivanildo Monteiro de Azevedo, Ana Paula L.F. Vasconcelos, Fernando Kenji Kamei and Silvio Romero de Lem, “Using CMMI together with agile software development: A systematic review”, *Elsevier*, 2015, p20-43.

[10] Craig Anslow and Frank Maurer, “An Experience Report at Teaching a Group Based Agile Software Development Project Course”, *ACM*, 2015, p1-6.

[11] Nauman Bin Alia,Kai Petersen and Kurt Schneider, Flow-assisted value stream mapping in the early phases of large-scale

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

software development", *Elsevier* , 2015, p1-15.

[12] Matthew B. Dwyer, "Connecting and Serving the Software Engineering Community", *IEEE*, 2016, Vol. 42, No. 3, 2016, p1-3.

[13] Akond Ashfaque Ur Rahman, Eric Helms, Laurie Williams, and Chris Parnin, "Synthesizing Continuous Deployment Practices Used in Software Development", *IEEE*, 2015, p1-10.

[14] Nauman Bin Ali, Kai Petersen and Breno Bernard Nicolau de Franca ,"Evaluation of simulation-assisted value stream mapping for software product development: two industrial cases", *Information and Software Technology* , 2015, p1-25.

[15] Tanja Suomalainen, Raija Kuusela and Maarit Tihinen, "Continuous planning: an important aspect of agile and lean development", Int. *J. Agile Systems and Management*, Vol. 8, No. 2, 2015, p1-31.

[16] Fabian Fagerholm , Marko Ikonen , Petri Kettunen , Jürgen Münch , Virpi Roto and Pekka Abrahamsson, "Performance Alignment Work: How software developers experience the continuous adaptation of team performance in Lean and Agile environments", *Elsevier* , 2015, p1-16.

[17] ChenYanga, PengLianga and ParisAvgeriou, "A systematic mapping study on the combination of software architecture and agile development", *Elsevier*, Vol. 111, 2016, p157-184.

[18] Umar Ruhi and Okhaide Akhigbe, "Lean Development in Design Science Research: Deliberating Principles, Prospects and Pitfalls", *Springer*, 2016, p286–300.

[19] Tim Dreesen,Robert Linden,Caroline Meures,Nikolaus Schmidt and Christoph Rosenkranz," Beyond the Border: A Comparative Literature Review on Communication Practices for Agile Global Outsourced Software Development Projects", *IEEE* , 2016, p1-10.

[20] Marco Kuhrmann and Jürgen Münch, "When Teams Go Crazy: An Environment to Experience Group Dynamics in Software Project Management Courses", *IEEE*, 2016, p1-10.

[21] Alex Osadchyy and Jon Webber, "A PRACTICAL MANAGEMENT SYSTEM FOR THE EFFECTIVE USE OF OFFSHORE SOFTWARE PROJECT OPPORTUNITIES", Review *of Business & Finance Studies, Vol.* 7, No. 1, 2016, p1-18.

[22] Minna Isomursu, Andrey Sirotkin,Petri Voltti and Markku Halonen , "User Experience Design Goes Agile in Lean Transformation – A Case Study", *IEEE*, 2012, p1-10.

[23] Abhinaya Kasoju , Kai Petersen and Mika V. Mäntylä, "Analyzing an automotive testing process with evidence-based software engineering", *Information and Software Technology*, Vol. 55 , 2013, p1237-1259.

[24] Shahabuddin, S. M. and Prasanth, Y, "Integration Testing Prior to Unit Testing: A Paradigm Shift in Object Oriented Software Testing of Agile Software Engineering", Indian Journal of Science and Technology, Vol. 9, No. 20, 2016, p1-10.

[25] Fraser, G., and Arcuri A, "Whole Test Suite Generation", IEEE, Vol. 39, No. 2, 2013, p1-16.

[26] Coggle, "*Coggle.it.* Available: https://coggle.it/. Last accessed 13th July 2017".

[27] Brown, A. W., Ambler, S., and Royce, W, "Agility at Scale: Economic Governance, Measured Improvement, and Disciplined Delivery. Software Engineering in Practice", IEEE, 2013, P873-881.

[28] W. Royce, "Software Project Management A Unified Approach", Addison-Wesley, 1998.

[29] Hana Rashied Ismaeel and Abeer Salim Jamil, "Software Engineering Cost Estimation Using COCOMO II Model", 2007, P1-26.

[30] Rachel Burger, "The Ultimate Guide to Project Management Mind Mapping", Retrieved from http://blog.capterra.com/ultimate-guide-project-management-mind-mapping/ , Accessed on 29 July 2017.