Reduction of Defect Generation in Software Projects using LSS Approach

Syed Mahiuddin[*], Jamal Uddin[*], Asim Zeb[†]

Abstract

A defected software usually couldn't perform its intended activities and client will not accept and pay for the product. Also, they can ask penalties. To make acceptable to clients, rework will require and more rework effort will make it costlier, profit margin will reduce. Less rework, more profit and improved client satisfaction are correlated and works in integrated form are directly proportional to each other. Lean and Six Sigma are used as process improvement tools in the software organizations. Lean Sigma is combination of Lean and Six Sigma process performance improvement frameworks and is more powerful compared to the individual is approaches. The delivery quality is measured regularly and defects reported was observed. High defect rate was observed during addition of new features or customization of the existing features of the product. This impacted deliverable quality and customer satisfaction both. To address these issues a process improvement initiative was initiated in the development phase with LSS as a tool and Define Measure Analysis Improvement Control (DMAIC) framework as the approach. As an initial step baseline performance was measured and it was 2.8 Sigma (Zst). The detailed data was used for identifying roots causes and formulating action items. Then, detailed defect data was collected and analyzed. Finally, performance level was measured after the initiative and it was 3.3 Sigma (Zlt). Accordingly, the defect generation rate was reduced by 50% which was previously observed as 40%. Therefore, the proposed approach helps in identifying defects, closing and preventing defect generation in early phase of software development.

Keywords: Lean, Six Sigma, Software defect, Software quality

Introduction

Most of our activities are now technology driven and computer programs are integral part of it. Starting from the microchip to any high-volume tasks are not operated by software driven way. Hence, software plays a pivotal role in the progress of our modern society. In the last few years there has been tremendous progress in the field of information technology and computing system (Berisha-Shaqiri 2014). Today Software programs are part of the production systems and helping in improving efficiency and reducing defects (Miles 2001). Also, software and internet is playing a key role in globalization (Douglas 2002). With the advancement of technology,

[*]Department of Physical & Numerical Sciences, Qurtuba university of science & Information Technology Peshawar, Pakistan. syedmahiuddin620@gmail.com

[†]Abbottabad University of Science & Technology, Abbottabad, Pakistan

the software development process has also been changed over the last decades. Now most of the software development follows Agile model rather than the traditional waterfall model (Hoda, Salleh et al. 2017), (Takagi and Daisuke 2007).

However, with the advancement of technology, availability of skill and competency the software business also facing competency challenges from developing countries and the startups (Wang and King 2000). To be in the business they need to be competitive in the market keeping the delivery quality high to be the first preference of the customers by improving their delivery quality (Mishra and Alok 2009). In order to achieve the same, they are now focusing to reduce the defects in the deliverables. It will improve the product quality and at the same time it will reduce rework percentages which is currently around 30% (Summers 2013). The reduction of the rework will reduce the cost of production and make the offering more competitive. This clearly signifies that there is a room for improvement in deliverable quality of software. The software quality can be improved by either improving the development process or by doing some value addition in the overall development process (Conboy and Morgan 2011).

Motorola was trying to improve the quality of their products in early 90's developed one quality improvement framework (Breyfogle and Forrest 1999). This framework was data driven (i.e. using historical data for solving the problems) and based on the principals of Statistical Quality Control (SQC). This framework named as Six Sigma was applied to business processes and it gave great result. Later, different companies started following this framework and most of them were benefitted. Now Six Sigma is used across the industries for improving performance by making the process more effective and robust. The limitations of Six Sigma include difficult implementation, complications, may get costly in the long run. While limitations of Lean include inability to effectively calculate projected (Return on Investment) ROI. Lean Methodologies do not account for the whole system and dynamic behavior.

The integration of LSS has ample evidence of successes in industry as well as in educational institutions in different parts of the world. LSS (LSS) procedure is implemented in Indian higher education system and the results were expeditious (Mukhopadhyay 2017). LSS is an effective methodology to maximize stakeholders' value by improving quality, efficiency, customer satisfaction and optimizing costs. The LSS implementation had a great positive impact on the bottom-line performance i.e. the financial performance of the organization (Lee, Tai et al. 2013).

Considering these aspects, it was thought to use Lean principals along with the Six Sigma principals in case of application software development. Software companies engaged in software development and maintenance are facing defect lickage problem to the client. The events started complaining about quality of deliverables. They then thought to improve client satisfaction by improving deliverable quality through defect reduction. An improvement initiative will be initiated following LSS methodology Define Measure Analysis Improvement Control (DMAIC) framework was selected for this purpose. and data was collected after four months of implementation.

The aim of this research is to use integrated framework of Six Sigma and Lean principals to reduce defect generation and improve the quality of deliverables for software projects. It was also reported that 40% defect reduction was possible using Six Sigma principals (Roy and Samaddar 2015). Hence, the present research aims integrate Lean with Six Sigma to reduce defect generation rate further, at least 50%.

The rest of the paper is organized as follows: The previous work related to this topic is summarized in Section II. The research methodology used in this study is described in Section III. Experimental design of this approach is described in Section IV. The results are presented in section V and the detailed analysis of this approach is presented in section VI. The Conclusion of the paper with future directions are explained in section VII and section VIII.

Background Study

LSS can be successfully used to improve such processes with some limitations. The limitations include issue with data handling, exception handling, dealing with missing value and analysis of unstructured data. Though we are talking about the applicability of LSS across the industry, practically it is not (Antony 2014). In the education sector the readiness factors are awareness, willingness to improve, cost benefits, regulations etc. Presence of these factors helps in implementation of LSS in higher education (Galli 2018).

LSS is mainly used for reducing waste and cost to improve bottom-line performance. However, there are chances of introduction of new risks in the newly developed process which can have some adverse impact in the long run (Aomar 2017). Hence, implementation of organization wide LSS based on pilot outcome can introduced risks related to resource, data quality and project selection.

Various software improvement methodologies like TRIZ, Agile, Lean and Six Sigma were studied. TRIZ is framework for specific process improvement (Grace, Slocum et al. 2001). This framework has main 40 principals with specific solution approaches. If the problem falls within the domain, specific solution is there; otherwise no solution is available. On the other hand, Agile brought a new flavor in the traditional software development process by emphasizing on workable delivery.

Like other product defect is a concern for software also. There are several process improvement frameworks available for improving software processes. Software Process Improvement and Capability determination (SPICE) and Capability Maturity Model Integration (CMMI), International Standards Organization (ISO) are couple of widely used models (Shelpar 2013). These models have their own pros and cons, hence should be carefully selected for application.

A simple PDCA (Plan–Do–Study–Act Cycle) approach on historical data can reduce the defect generation in software application (Anees 2017). This is very simple process and lack of integrated data driven approach. This approach can bring defect level to a certain extent but defect reduction beyond a certain level is not possible.

IT service industry uses Control Objectives for Information and Related Technologies (COBIT), Information Technology Infrastructure Library (ITIL) methodology for managing the service operations i.e. IT governance and services respectively (Herrera 2019). Meta model based on the continuous improvement approach of IT services integrated with Lean methodology shown better result than the traditional frameworks. Still this concept is very new and need more application before generalization.

Defect prediction models are effective in predicting software defect. One approach to predict software defect is Probabilistic-ABC and feature selection (Kumar 2017). This can predict the software defects with large number of feature sets. This was used in a NASA data set and prediction was highly satisfactory. However, the positive outcome of this model can be used further to develop a dynamic prediction model.

Increase of interoperability of software increased demand for software defect prediction to reduce operational hazards (Changzhen 2019). This can be achieved through proper usage of dimensionality reduction process. Local Tangent Space Alignment (LTSA) algorithm found to be helpful for this purpose and its prediction accuracy is 1-4% higher compared to Support Vector Machine (SVM) and Locally Linear Embedding and Support Vector Machine (LLE-SVM). Also, deep learning-based algorithm Deep Belief Network to learn semantic based structured code representation automatically learns and successfully predict the trend of software defect generation on average by 14.7% in precision, 11.5% in recall, and 14.2% in F1 better comparing with traditional features (Wang and Taiyue 2016).

Software defect prediction can also be achieved through Principle Component Analysis (PCA) incorporating maximumlikelihood estimation for error reduction and neural network-based classification techniques for prediction (Jayanthi and Florence 2019). This was tested in NASA software dataset with MATLAB simulation tool. The outcome was quite good and better than other techniques in some cases.

In a review on application of Lean, Six Sigma and Agile metrology, it was concluded that these are effective and highly used in software organizations (Badwe and Erkan 2018). However, usage is not so much compared to manufacturing or service sectors. While other sectors are enjoying the benefits of these frameworks, software sector is lagging behind. It is recommended for the software organizations to uses Six Sigma as a process improvement framework and improves their overall business results, more the use better is the performance. In recent time the volume of data is very high with different types of data (Gunasekaran 2019).

The comparison of existing and previous work of LSS in the field of software development is presented in Table 1.

Table1: Comparative analysis of existing work of LSS in Software Development

LSS	**(Gijo and Antony 2019)**	**(Huang 2016)**	**Current Research**
Project Completion Duration	6 Months	3 Months	3 Months
Process Improvement	25%	47%	50%
Data Points	Before= 690 After=460	Before = 320 After= 280	Before= 1410 After=971
Defect Reduction	No	No	Yes
Projects Considered	1	1	3
Preliminaries	Limited	Limited	Detail
Analysis of LSS	Limited	Limited	Detail

Lean Six Sigma (LSS) Approach

Six Sigma well-defined methodology Six Sigma is a data driven methodology; data accuracy is highly important for Six Sigma. Six Sigma is all about improving business result through process improvement. Lean is the set of "tools" that assist in the identification and steady elimination of waste. As waste is eliminated quality improves while production time and cost are reduced. Lean Six Sigma is a method that relies on a collaborative team effort to improve performance by systematically removing waste and reducing variation.

An integrated approach of Lean and Six Sigma (LSS) is used to improve the software development process performance by reducing the defect rate. The proposed research process steps are presented in Figure 1 and are described as below;

Step 1**:** Identification of research problem – In this step based on the literature review and other available information identification and finalization of research problem is done.

Step 2**:** Initial data collection – In this step initial data collection is done to find out the performance status before application of LSS.

Step 3**:** Data Processing – This is the third step of the overall process. The objective of this step is to remove the outliers and make the data ready for initial analysis purpose.

Step 4*:* Data Analysis **-** This is the next step after data processing. In this step data analysis is done to understand the (a) Baseline Performance (b) Root Causes of the Low Performance

Step 5: Action Item Identification – This step will be followed by the Root Cause Analysis. The objective of this step is identification of action items to address the negative impacts which are causing low performance.

Step 6*:* Action Item Implementation **-** Once action items are identified implantation plan will also be developed. These action items will be implemented, which are practically feasible to implement.

Step 7**:** Data Collection (after implementation) **-** After running the new process for some time, data will be collected, and processed for removal of outliers.

Step 8**:** Performance Measurement (after) – In this step, after the data processing, process performance will be calculated to check if these is improvement in performance after implementation of LSS.

Step 9: Process Standardization_ Once performance analysis is done and satisfactory result is observed, the new process will be standardized. All process manuals, process aids and standard operating procedures will be updated.

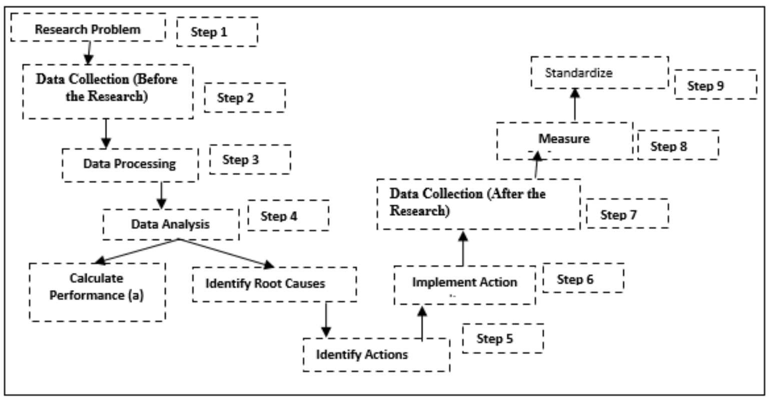


Figure 1: Proposed Research Process

Project charter

(i) All projects are related to capital market business domain.
(ii) Techno log used_ Java.
(iii) Database_ orade 11 G.
(iv) Front End _HTHL 5.
(v) Effort _ 30 man months to 60 man months.
(vi) Team structure_ Project Manager, Business Analyst, Developer testers

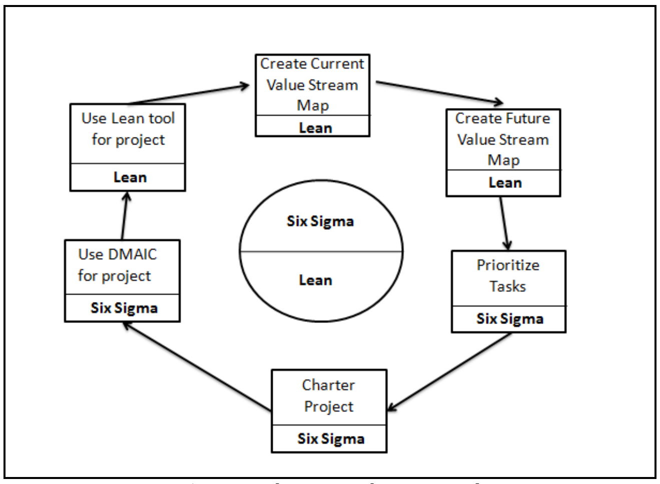


Figure 2: The Research Framework

LSS performance improvement initiative followed DMAIC framework. Testing finding are captured in defect tracking tool JIRA. Defect records downloaded from this tool of companies like (IBM, HCL and Tata Consultancy Services). The data set containing parameters like Test case ID, Test case description, expected result, Actual result, Test case status (pass/ fail), Finding category (Bug/New feature/Enhancement), Finding Type (Blocker/Major/Minor), Root Cause and Finding status (Open/ Close/Ignore)

The records are tracked till closure and kept in the system for future references. Based on the approval from the senior management, required data dump was provided internally for the use for three customization and deployment projects for three different companies. In the defect data there are different variable. The variables which are required for the performance improvement projects includes; Defect Type – Bug / Improvement, Defect Category - Technical Error/ GUI Issue, Defect Status – Fixed / Ignore and Defect Root Cause- Incomplete requirement/ Domain

Knowledge/ Logic Implementation/ Impact of Bug Fix. In this paper, attempt was made to include more projects to make the outcome more generic hence more three projects were added.

Results and Discussion

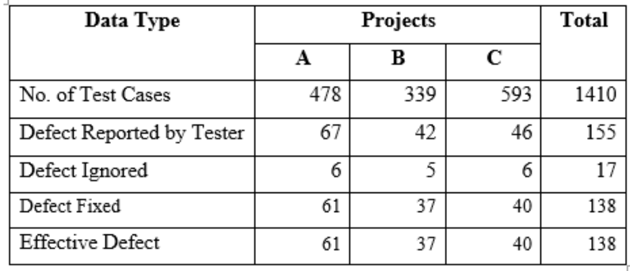
LSS initiative started for three projects having requirements at three different companies named IBM, HCL and Tata Consultancy Services. Based on the client requirements, software is customized to make it fit for use in different companies. There are lots of requirements in a customization project. The requirements are of two types; new development and change in the existing functionality. Systematic approach is essential for improving performance. This initiative was driven using DMAIC framework and some Lean tools have been used. The Six Sigma concepts used in this process are; Process Diagram, Sigma Scale and tools like brain storming, data collection, pareto chart, hypothesis testing.

The Lean concepts those used in this initiative are; Mistake proofing – introduction of impact analysis in audit checklist to make it mandatory, VA/NVA Analysis- Based on this analysis, many steps in the review process was removed to ensure more time availability for the developer. Two phase testing is done; functional testing for validating the working of requirements and automation testing as the regression testing for overall functionalities. In all projects test cases and test execution records are maintained in JIRA. To validate the requirements or the overall functionalities, test cases are prepared. Single test cases are written to test single functionality.

Defect Analysis before applying LSS

The defect distribution of two different type of activities are summarized in Table 2.

Table 2: Summary of Defect (Before applying LSS)



Data Type	Projects			Total
	A	B	C	
No. of Test Cases	478	339	593	1410
Defect Reported by Tester	67	42	46	155
Defect Ignored	6	5	6	17
Defect Fixed	61	37	40	138
Effective Defect	61	37	40	138

By the help of table, we can calculate and judge the total data. The number of cases is 478 in project A and in project B, the number of test cases were less then project A. In project C it is higher among all the projects because it is 593 and hence the total number of cases were recorded 1410. But the defect reported by tester were 155 in total and the higher defect reported by tester were in project A that is 67. Defect ignored was recorded equally in project A and in project C. It is same in number but project B defect ignored were less and total defect ignored were 17. Project A fixed defect is higher it is 61 and in project B it is 37 but project C defect fixed is higher than project B it is 40 and total is 138. Therefore, effective defect is same like defect fixed it is also 138 in total. Moreover, defect summary based on the origin is given in the Table 3.

Table 3: Defect Summary Origin (Before applying LSS)

Defect Origin	Defect Count	Defect in %
Customization	89	64.49
New Development	49	35.51

It was observed that in total of 138 defects in which 89 defects (64.49 %) contributed by customization projects and 49 defects (35.51%) contributed by new development. It can also be seen that new development has less defects as compared to customization.

Process Performance Level (Baseline)

In Six Sigma term, process performance levels are:
Long term (Z_{LT}) and Short Tern (Z_{ST}) = 1.5 + Z_{LT}
In the present initiative, defect count 138, total test cases i.e. the opportunity is 1410 and opportunities per unit is 1.
Defect Rate = (Total Defect/ Total Test Cases) = 0.1
Defect per Million Opportunities = (Defect count *1000000/ Total Test Cases) = 97872
Using the formula,
Sigma Level = ABS (NORMSINV(DPMO/1,000,000))) we get,
Long term $(ZLT) = 1.3$, Short term $(ZST) = 2.8$

Identified action items were implemented to close the defects. A detailed analysis of the defect cause is summarized in Table 4 and corresponding Pareto Chart is given in Figure 3. The records and defect causes are measured and summarized in Table 4. It was observed that, out of total 138 defect, 33 defects caused due to lack of proper domain knowledge of the development team members, in 21 cases requirement clarity was missing and lead to defect generation. One of the main issues in software development is the tight schedule. In present case 26 defects generated due to tight timeline.

Table 4: Defect Summary by Causes

Defect Cause	Defect Count
Domain Knowledge	33
Impact Analysis	48
Unclear Requirement	21
Crunch Timeline	26
Others	10
Total	138

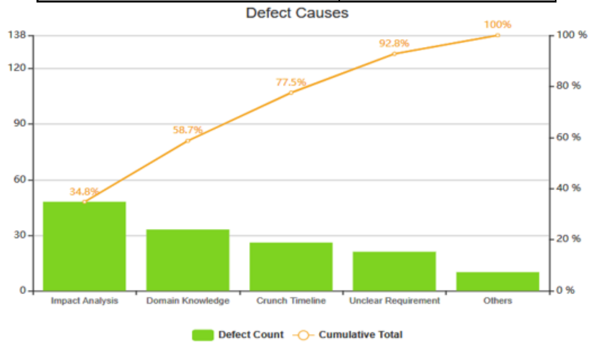


Figure 3: Pareto Analysis of Defects

To identify major causes of defects of Pareto analysis was conducted. The objective was to identify 20% of causes which are responsible for 80% defects. The Pareto analysis reveals that top causes contributing 80% defects are; Impact Analysis (34.8%), Domain Knowledge (24%) and Crunch Timeline (19%). Now LSS is used to identify the exact causes of defect generation and probable action items to address the same. The brainstorming sessions and multiple focus group discussions conducted with the project team members. The discussion summary is given in Table 5.

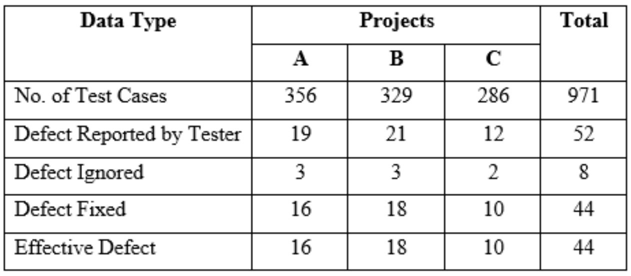
Defect Analysis after applying LSS

A new initiative started to improve the level of domain knowledge of the employees. Initially small sessions within the project teams started for the different parts of the products. Also, development of e-learning courses is in progress to make this domain knowledge training hassle free and easily accessible for all the employees. After four months of implementation of these action items defect data again collected from another three projects. The defect data are summarized in Table 6. It presents the project defect after the implementation of the action items as identified in the analysis phase. Here, total 971 test cases were available (Project A= 356, Project B= 329 and Project C= 286). Total 52 defects were logged by the testers (Project A= 19, Project B= 21 and Project C= 12) and 8 defects were ignored by the project teams (Project A= 3, Project B= 3 and Project C= 2). Hence, total effective defects after the LSS Project was 44 (Project A= 16, Project B= 18 and Project C= 10).

Table 5: Defect Root Causes

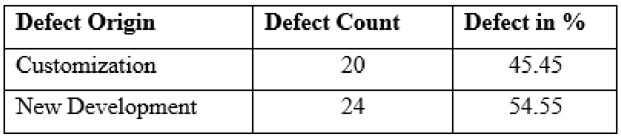
Defect Cause	**Sub Causes**	**Action item to address the cause**	**Remark**
Impact Analysis	Impact analysis plays an important role in identifying the impact of change in other software functionalities. It was observed it is not done always, hence impact remain unknown to the developers and sometime cause new defect	Make impact analysis mandatory for the customization requirements to ensure the impact of change in rest of the functions.	Change in process definition is required to ensure the same
Domain Knowledge	This domain is niche and it was observed the knowledge is limited within a group of SMEs. Inadequate knowledge sometimes leads to defect generation.	Prosper knowledge sharing to be done to improve domain knowledge of the workmen.	Change in training manual is required to make Domain training mandatory for all the employees. Depth of the training will depend on the role of the employee.
Crunch Timeline	It was observed that, there was excessive review before the delivery. Due to which developers was getting less time than the required time, which hampering the quality of deliverable.	The numbers of review to be reduce to give more time to the developers for their work.	Change in the review process is required to cater this requirement

Table 6: Summary of Defect (After applying LSS)



Data Type	**Projects**			**Total**
	A	**B**	**C**	
No. of Test Cases	356	329	286	971
Defect Reported by Tester	19	21	12	52
Defect Ignored	3	3	2	8
Defect Fixed	16	18	10	44
Effective Defect	16	18	10	44

Table 7: Defect Summary Origin (After Applying LSS)



Defect Origin	Defect Count	Defect in %
Customization	20	45.45
New Development	24	54.55

From Table 7, it was seen and observed through different analytical studies that in total of 44 defects that studied which 20 defects (45.45%) contributed by customization projects and 24 defects (54.55%) contributed by new development. We can see that new development has less defects as compared to customization.
Total test case= 971, total defects =44 and defect opportunities per unit =1
Defect Rate = Total Defects/ Total Test Cases = 0.05
Defect per Million Opportunities = (Defect count *1000000/ Total Test Cases) = 45314
Using the formula, Sigma Level = ABS (NORMSINV(DPMO/1,000,000))) we get,
Long term (Z_{LT}) = 1.7 and Short Tern (Z_{ST}) = 3.2

The correct output rate before Two tailed proportion test was conducted to ensure if correct output is higher after the implementation of LSS. The outcome of the test at 95% confidence level suggests rejection of the null hypothesis. Therefore, there is enough evidence to applying LSS proportion p1 is less Correct output rate after applying LSS than p2, at the 0.05 significance level. Which signifies that percentage of correct output is higher after implementation of LSS.
Improvement of Defect Rate = (0.1-0.05) *100/0.1 = 50% accordingly we achieve the goal (50 %) of the initiative.

Conclusion and Future Work

In the pervious research, only Six Sigma was used but in this work LSS was used as improvement techniques LSS is more powerful than Six Sigma, so accuracy improved due to this. Whereas, in the present research, Lean was used along with Six Sigma for the same objective of defect reduction in software development. At the beginning the performance level was Zst (2.8) Sigma. Then LSS tools and techniques were used to identify the root causes of defects, Appropriate action were taken to address the problem and performance level improved to Zst (3.2) Sigma. Also defect generation rate reduced by 50% which is quite good for quality and rework point of view. Hence it can be concluded that LSS is more effective in reduction of software defect generation than Six Sigma. This LSS study was conducted in financial domain of the software development. This was critical, sensitive and confidential project. To improve the performance some process related changes were made to the existing processes. This LSS project was conducted in the software development process which is not very common. This initiative was for customization and deployment of the software. The projects under the study were development and enhancement in nature. Also, these projects are very specific to the capital market domain. Hence, more study is required to generalize the result.

References

Anees, A. (2017). "Software process improvement models and their comparison." *International Journal of Advanced Research in Computer Science* 8(5): 200_213.

Antony, J. (2014). "Readiness factors for the Lean Six Sigma journey in the higher education sector." *International Journal of Productivity and Performance Management* 5(7): 1-12.

Aomar, M. (2017). "Reducing the interruption of power distribution:A Six Simga application." *2nd International Conference on knowledge Engineering and applications (ICKEA) IEEE.*

Badwe, S. and T. E. Erkan (2018). "A Taxonomy of Lean Six Sigma and Agile Methodologies used in Software Development." *International Journal of Engineering Research and Technology,* 11(7): 725-753.

Berisha-Shaqiri (2014). "Impact of information technology and internet in businesses." *Academic Journal of Business, Administration, Law and Social Sciences IIPCCL Publishing,* 1(1): 1_7.

Breyfogle, F. and T. Forrest (1999). Implementing six sigma part 1. *The Quality Management Forum*, ASQ, Summer.

Changzhen, Z. (2019). "Establishing a software defect prediction model via effective dimension reduction." *Information Sciences* 477: 399-409.

Conboy, K. and L. Morgan (2011). "Beyond the customer: Opening the agile systems development process." *Information and Software Technology* 53(5): 535-542.

Douglas, K. (2002). "Theorizing Globalization." *Sociological Theory* 20(3): 285-305.

Galli, B. (2018). "Can project management help improve lean six sigma?" *IEEE Engineering Management Review* 46(2): 55-64.

Gijo, E. and J. Antony (2019). "Application of Lean Six Sigma in IT support services–a case study." *The TQM Journal.*

Grace, Slocum and Clapp (2001). "A new TRIZ practitioner's experience for solving an industrial problem using ARIZ 85C." *The TRIZ Journal.*

Gunasekaran (2019). "Big data in lean six sigma: a review and further research directions." *International Journal of Production Research: 1-23.*

Herrera, M. (2019). Using metamodeling to represent lean six sigma for IT service improvement. 2019 *IEEE 21st Conference on Business Informatics (CBI), IEEE.*

Hoda, R., N. Salleh, J. Grundy and H. M. Tee (2017). "Systematic literature reviews in agile software development: *A tertiary study." Information and Software Technology* 85(15): 60-70.

Huang (2016). *Application of lean six sigma methodology in software continuous integration*. Key Engineering Materials, Trans Tech Publ.

Jayanthi and L. Florence (2019). "Software defect prediction techniques using metrics based on neural network classifier." *Cluster Computing* 22(1): 77-88.

Kumar, R. (2017). A novel probabilistic-ABC based boosting model for software defect detection. 2017 *International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), IEEE.*

Lee, K.-l., C.-T. Tai and G.-J. Sheen (2013). "Using LSS to improve the efficiency and quality of a refund process in a logistics center." *International Journal of Lean Six Sigma.*

Miles, P. (2001). "Globalisation–Economic Growth and Development and Development Indicators." *Planet Papers* 4(3): 210_218.

Mishra and Alok (2009). "Software process improvement in SMEs: A comparative view." *Computer Science and Information Systems* 6(1): 111-140.

Mukhopadhyay, K. (2017). Application of Lean six sigma in Indian higher education system. 2017 *International Conference on Innovative Mechanisms for Industry Applications (ICIMIA),* IEEE.

Roy, S. and S. Samaddar (2015). "To Reduce Deffect in Software Development: A Six Sigma Approach." *Center for Quality* 12(6): 345_352.

Shelpar, M. (2013). "“Software Process Improvement Model”." *International Journal of Advanced Research in Computer Science and Software Engineering,* Vol. 3(6): 313-315.

Summers (2013). "Effective Method for Software and System Integration."_*Taylor& Francis Group.* 12(8): 28_29.

Takagi and Daisuke (2007). "Innovation in software development process by introducing Toyota Production System." *Fujitsu Sci. Tech. J* 43(1): 139-150.

Wang and Taiyue (2016). Automatically learning semantic features for defect prediction. 2016 IEEE/ACM *38th International Conference on Software Engineering (ICSE), IEEE.*

Wang, Y. and G. King (2000). *Software engineering processes: principles and applications,* CRC press.