ISBN 978-981-14-4787-7
Proceedings of 2020 the 10th International Workshop on Computer Science and Engineering (WCSE 2020)
Yangon (Rangoon), Myanmar (Burma), February 26- February 28, 2020, pp. 27-35
doi: 10.18178/wcse.2020.02.006

Comparative Results of Dependent and Independent Variables Focused on Regression Analysis Using Test-Driven Development

Myint Myint Moe[1+], Khine Khine Oo[2]

[1]University of Computer Studies, Yangon, Myanmar
[2]University of Computer Studies, Yangon, Myanmar

Abstract. Test-Driven Development is a software engineering technique and should be tested previous the code, which make sure to read the unit test. The goal of this paper is to examine product quality and programmer productivity on the number of tests for the consequence of test-driven development. This system builds the acceptance test suite metric and the ordinary least squares method of regression analysis to assess the impact of the process on dependent variables and independent variables. The results of this paper are that if developer productivity is the actual effect, external code quality will be fewer decreased and external code quality is the actual effect if developer productivity will be fewer reduced. TDD responds to assist the delivery of high-quality products both operational and technical perspective while enhancing developers' productivity. TDD leads to less defects and fewer debugging period which actual code can be assured by writing tests first and thus serving the developer get a finer understanding of the software requirements. This proposed system evaluates the ordinary least squares and the acceptance test suite metric of regression analysis based on a fixed time-frame.

Keywords: Test-Driven development, Unit test, no: of tests, External Quality, Developer Productivity

1. Introduction

Test-driven development is the foundation of software development but it responds unit tests previous production code. Test-driven development is part of the agile code development approach and drive from Extreme Programming and the Agile Platform. Before the code development, developers encourage to compose tests [1- 2]. The possible of TDD describes various positive effects. TDD isn't a testing approach, yet rather a development and design method in which the tests are composed before the production code. During the implementation, the tests are added step by step and when the test is passed, the code is refactored to improve the inside structure of the code, without changing its outside behavior. TDD cycle is repeated until the whole functionality is implemented. For each little function of an application, TDD begins with designing and developing tests. First, the test is created that distinguishes and approves what the code will do in the TDD approach. Make the code and after that test in the typical testing process. The developer can be self- assurance that code refactoring is not destroyed any existing functionality for re-executing the test cases [3- 4].

This paper is structured as follows. The issue of Test-Driven Development initiated in Section (1). The obviousness of the number of tests, external code quality and, developer productivity on test-driven development (TDD) expressed in Section (2). A background of the whole procedure of test-driven development is presented in Section (3). Section (4) describes the contribution of the interrelation of the number of tests, external code quality, and developer productivity. Next, Section (5) shows the proposed system of this paper. Section (6) in this paper describes discussion and comparison of results. Finally, the conclusion of this paper describes in Section (7).

[+] Corresponding author. Tel.: +09448018175
E-mail address: myintmyintmoe.ucsy.1971@gmail.com.

2. Objectives

One of the approaches of software progression was test-driven development. In recent years, this approach has become familiar in the industry as a requirements specification method. TDD is intended to make the code clearer, simple and bug-free [4- 5]. The goal of the proposed system analyzes the consequence of dependent variables and independent variables on TDD**.** It observes the nature of the correlation between the number of tests (TEST) and external code quality (QLTY), and the correlation between the number of tests (TEST) and developers' productivity (PROD). The good points of Test-Driven Development upgrade software quality and accelerate the testing process. This approach more productive program code and make fewer efforts per line of code. By decreasing code complexity supporting, the proposed system validates the exactness of all codes and allows developers assurance. It is used persistently over time and motivates developers to create higher code quality. This system assures the correctness of the code and helps developers' gain a better understanding of the software requirements which leads to fewer defects and less debugging time. This intends more productive and make fewer efforts per line of code.

3. Background Theory

TDD builds up the early development of tests, at the time changes are welcomed and advanced with functional components. So, correcting defects is made earlier in the process. Test-Driven Development is a coding technique. Kent Beck (inventor of Extreme Programming and JUnit) invents Test-Driven Development refers to a style of programming where three activities are closely interlinked. Three activities are Coding, Testing, and Design. At first, its key idea is to execute initial unit tests for the code, must be implemented, and then implement the actual feature of it. One of the features of software system requirement is tackled subtask or user stories, which are designed to easily express and understand. These can be easy to change by the end-user as they like during the project's handle time [5-7].

3.1. Test-Driven Development Process

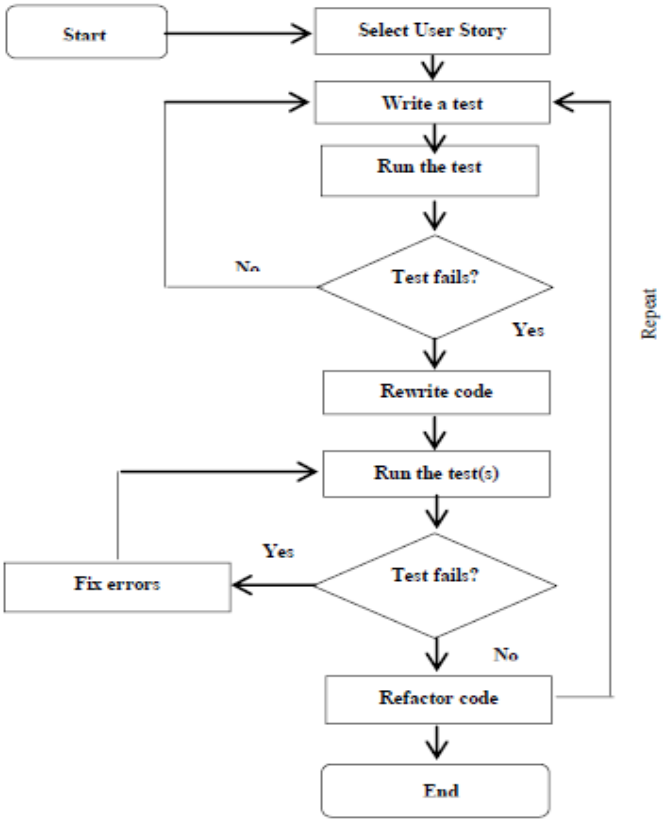

Fig. 1: Test-Driven Development flow

The TDD action is introduced in Figure 1, and includes the following steps:

(1) Select a user story,

(2) Write a test that fulfills a small task of the user story and run this test. Then produces a failed test,

(3) Re-write the production code necessary to implement the feature,

(4) Execute the pre-existing tests again, where any failed test is existed. When the code is true effectively and finally goes to the refactoring stage.

(5) When the refactoring stage is completed, the actual production code is manufactured and the user can select a new user story again. This method constructs some benefits, focus on the responsibility of increasing the quality of the software product and the productivity of programmers.

4. Contribution

The contribution of this paper observed the number of tests (TEST), external code quality (QLTY) and developer productivity (PROD). The number of tests is measured by the count of a single JUnit test case. External code quality is proposed the percentage of acceptance tests passed for the implemented user stories. The developer productivity is proposed the percentage of implemented user stories.

5. Proposed System

In this proposed system, the acceptance test suite metric and the ordinary least squares method of regression analysis apply to measure the number of tests, external code quality, and developer productivity.

5.1. Research Questions

This system concentrates to evaluate two outcomes on the following system: external code quality based on the number of tests and developer productivity based on the number of tests.

RQ1 (RQ-QLTY): Does a higher number of tests indicate higher quality?

RQ2 (RQ-PROD): Does a higher number of tests indicate higher developer productivity?

The notion of external code quality in RQ-QLTY and productivity in RQ-PROD are based on the acceptance test suite metric and the ordinary least squares method of regression analysis.

5.2. Method of Acceptance Test Suite Metric

In the proposed system, the acceptance test suite metric is used by analyzing to explore possible interactions such as number of tests, external code quality, and developer productivity. The acceptance test suite metric is a form of mathematical regression analysis [7-9]. Regression analysis is used to investigate the relationship between two or more variables and estimate one variable based on the others. Regression analysis is a powerful statistical method that allows for analyzing the relationship between two or more outcome variables of interest. QLTY and PROD are the dependent variables. TEST is the independent variable. QLTY defined as the percentage of acceptance tests passed for the implemented tackled tasks. PROD measured as the percentage of implemented tackled tasks. Table 1 provides the raw data used in the assessment. To calculate this low-level metric, this system uses an automated tool. The limited-time necessary to complete the task had an impact on the metric. In regression analysis, dependent variables are established on the vertical y-axis, while independent variables are established on the horizontal x-axis.

5.2.1. Number of Tests (TEST)

Number of Tests (TEST) is defined as numbers of JUnit assert statements within the unit test suite written by the participants while tackling the task. The numbers of test development as a single JUnit assert statements. TEST assessed by the count of the JUnit test cases. TEST is a ratio measure in the range [0, ∞]. The formula for calculating TEST is defined as [9]:

$$\text{TEST= no: of subtasks out of results from the no: of input subtasks} \quad (1)$$

$$\text{TEST = the numbers of JUnit assert statements within the unit test suite}$$

5.2.2. External Code Quality

The metric for external quality QLTY based on the number of tackled subtasks *(#TST*) for a given task. A subtask as tackled assesses if at least one assert statement in the acceptance test suite associated with that subtask passes. QLTY is a ratio measure in the range [0,100].

The number of tackled subtasks (*#TST*) is defined as:

$$\#TST = \sum_{i=0}^{n} \begin{cases} 1 & \#Assert_i(Pass) > 0 \\ 0 & otherwise \end{cases} \quad (2)$$

#TST = the number of tackled subtasks

n = the total number of subtasks

The formula for computing QLTY is defined as [10]:

$$QLTY = \frac{\sum_{i=0}^{\#TST} QLTY_i}{\#TST} \times 100 \quad (3)$$

$QLTY_i$ = the quality of the i^{th} tackled subtask

Where $QLTY_{i\ is}$ the quality of the i^{th} tackled subtask and $QLTY_i$ is defined as:

$$QLTY_i = \frac{\#Assert_i(Pass)}{\#Assert_i(All)} \quad (4)$$

#Assert$_i$ (Pass) = the number of JUnit assertions passing in the acceptance test suite associated with the i^{th} subtask

#Assert$_i$ (All) = the total number of JUnit assertions in the acceptance test suite associated with the i^{th} subtask

For example, supposing that a person assesses thirteen tackled subtasks (*#TST* = 13), this means that there are thirteen tackled subtasks for which at least one assert statement passes in the test suite. Let us assume that the acceptance test of the first analyzed tackled task contains 3 assertions, out of which three are passing. The acceptance tests of the fourth tackled task contain 10 assertions, of which three are passing and so on.

i.e. $(QLTY_4 = \frac{\#Assert_4(Pass)}{\#ASSERT_4(All)} = \frac{3}{10} = 0.3)$

$(QLTY = \frac{\sum_{i=1}^{\#TST} QLTYi}{\#TST} \times 100 = \frac{1+1+1+0.3+1+1+1+1+1+1+1+1+1}{13} \times 100 = 95)$

Table 1: Summary of acceptance tests used to calculate the metrics of Bowling Scorekeeper data-sets [10].

Task	Test	Assert
T1	3	3
T2	3	3
T3	2	2
T4	3	10
T5	5	5
T6	6	6
T7	8	8
T8	5	5
T9	5	5
T10	4	4
T11	2	2
T12	3	3
T13	2	2

Table 2: Solution of QLTY

Task	Test	Assert	QLTY
T1	3	3	1
T2	3	3	1
T3	2	2	1
T4	3	10	0.3
T5	5	5	1
T6	6	6	1
T7	8	8	1
T8	5	5	1
T9	5	5	1
T10	4	4	1
T11	2	2	1
T12	3	3	1
T13	2	2	1
	51	**58**	**95**

5.2.3. Productivity

The productivity metric (PROD) describes the amount of work successfully performed by the subjects. PROD is a ratio measure in the range [0,100]. The metric of PROD is calculated as follows [10]:

$$\text{PROD} = \frac{\#Assert(Pass)}{\#Assert(All)} \times 100 \quad (5)$$

For example, assume a person implements a tacked task with a total of 58 assert statements in a test suite. After running the acceptance test suite against the person's solution, 51 assert statements are passing.

i.e. $(\text{PROD} = \frac{\#Assert(Pass)}{\#Assert(All)} \times 100 = \frac{51}{58} \times 100 = 88)$

5.2.4. Assessment

The image below is a scatter plot**.** Scatter plots are used when this paper want to show the relationship between two variables. Scatter plots are called relationship plots because they show how two variables are interrelated. This analytical tool is most often used to show data correlation between two variables. This system expects that the regression analysis of the information compiled from the external code quality over the number of tests by TDD responds positively to questions RQ1. In the same way, this system expects that the regression analysis of the information compiled from the developer productivity over the number of tests by TDD responds slightly decrease to questions RQ2. In figure 2, the external code quality over the number of tests is improved by measuring the acceptance test suite metric of quality (QLTY). In figure 3, the developer's productivity over the number of tests is slightly decreased by measuring the acceptance test suite metric of productivity (PROD).

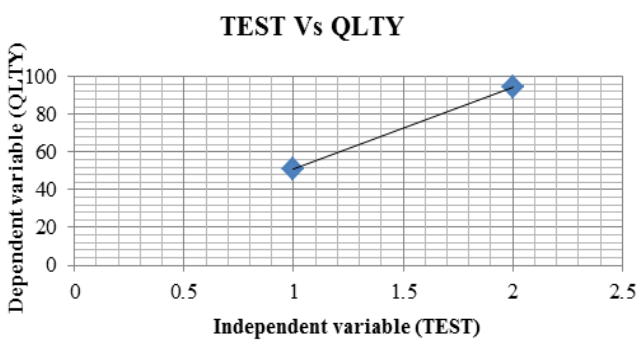

Fig. 2: QLTY is on the function of TEST

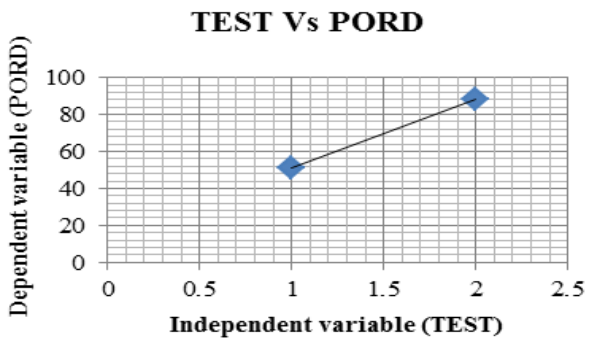

Fig. 3: PROD is on the function of TEST

5.3. Method of Ordinary Least Squares

QLTY and PROD are the dependent variables. TEST is the independent variable. The data set consisting of TEST, QLTY and PROD attributes were analyzed to discover outliers using both z-score and modified z-score methods [11-12]. Table 1 provides the raw data used in the assessment. In the proposed system, the ordinary least squares method is used by analyzing to explore possible interactions such as number of tests, external code quality, and developer productivity. TEST assessed by the count of the JUnit test cases. This is a ratio variable within the range [0, ∞]. QLTY explicated as the percentage of acceptance tests passed for the implemented stories. PROD measured as the percentage of implemented stories. The ordinary least squares method is a form of mathematical regression analysis used to determine the line of best- fit for data points. Each point of data describes the association between a known independent variable and an unknown dependent variable. In regression analysis, dependent variables are depicted on the vertical y-axis, while independent variables are depicted on the horizontal x-axis. The line of best-fit determined from the least-squares method has an equation that states the user story of the correlation between the data points. Line of best-fit equations may be determined by computer software models, which include a summary of outputs for analysis, where the coefficients and summary outputs explain the dependence of the variables being tested. The b_1 is the slope of the regression line. Thus this is the amount that the Y variable (dependent) will convert for each 1 unit convert in the X variable. The b_0 is the intercept of the regression line with the y-axis. Y-hat = $b_0 + b_1(x)$ is the illustrative regression line. This paper must determine b_0 and b_1 to draw this line. Y-hat is the predicted value of Y, and it can be obtained by plugging an individual value of x into the equation and calculating y-hat.

The ordinary least squares of regression analysis are computed as the formulas:

$$\text{Mean of TEST } (\bar{X}) = \frac{\Sigma X}{N} \quad (6)$$

ΣX = sum of all the individual #TEST data set
N = total number of # TEST data set

$$\text{Mean of QLTY (or) PROD } (\bar{Y}) = \frac{\Sigma Y}{N} \quad (7)$$

ΣY = sum of all the individual QLTY or PROD data set
N = total number of QLTY (or) PROD data set
Predicted value of Y,

$$b_0 = \hat{y} - b_1 x = \text{mean } (\bar{Y}) - b_1 * \text{mean } (\bar{X}) \quad (8)$$

b_0 = the intercept of the regression line with the y-axis

$$b_1 = \frac{\Sigma(X - \bar{X})(Y - \bar{Y})}{\Sigma(X - \bar{X})^2} \quad (9)$$

b_1= the slope of the regression line

$$\hat{y} = b_0 + b_1 x \quad (10)$$

$\hat{y}$= the sample regression line

The table-1 dataset consisting of TEST, QLTY and PROD attributes was tested to find outliers using both z-score and modified z-score method**s.** This paper used the dataset from [12].

Table 3: Dataset Used in the Assessment

TEST	QLTY	PROD
16	69	100
10	28	46
14	49	92
17	72	100
14	78	92
17	75	100
25	60	69
11	69	85
6	26	46
5	43	31
14	68	100
13	86	100
13	68	85
8	11	46
11	75	54
10	55	85

Table 4: Data of Computation for Correlation of TEST and QLTY

X (TEST)	Y (QLTY)	$X - \bar{X}$	$Y - \bar{Y}$	$(X - \bar{X})^2$	$(X - \bar{X})(Y - \bar{Y})$
16	69	3.25	10.69	10.56	34.73
10	28	-2.75	-30.31	7.56	83.36
14	49	1.25	-9.31	1.56	-11.64
17	72	4.25	13.69	18.06	58.17
14	78	1.25	19.69	1.56	24.61
17	75	4.25	16.69	18.06	70.92
25	60	12.25	1.69	150.06	20.67
11	69	-1.75	10.69	3.06	-18.70
6	26	-6.75	-32.31	45.56	218.11
5	43	-7.75	-15.31	60.06	118.67
14	68	1.25	9.69	1.56	12.11
13	86	0.25	27.69	0.06	6.92
13	69	0.25	10.69	0.06	2.67
8	11	-4.75	-47.31	22.56	224.73
11	75	-1.75	16.69	3.06	-29.20
10	55	-2.75	-3.31	7.56	9.11
204	**933**	**0.00**	**0.00**	**351.00**	**825.25**
$\bar{X}$=**12.75**	$\bar{Y}$=**58.31**				

Table 5: Data of Computation for Correlation of TEST and PROD

X (TEST)	Y (PROD)	$X - \bar{X}$	$Y - \bar{Y}$	$(X - \bar{X})2$	$(X - \bar{X})(Y - \bar{Y})$
16	100	3.25	23.06	10.56	74.95
10	46	-2.75	-30.94	7.56	85.08
14	92	1.25	15.06	1.56	18.83
17	100	4.25	23.06	18.06	98.02
14	92	1.25	15.06	1.56	18.83
17	100	4.25	23.06	18.06	98.02
25	69	12.25	-7.94	150.06	-97.23
11	85	-1.75	8.06	3.06	-14.11
6	46	-6.75	-30.94	45.56	208.83
5	31	-7.75	-45.94	60.06	356.02
14	100	1.25	23.06	1.56	28.83
13	100	0.25	23.06	0.06	5.77
13	85	0.25	8.06	0.06	2.02
8	46	-4.75	-30.94	22.56	146.95
11	54	-1.75	-22.94	3.06	40.14
10	85	-2.75	8.06	7.56	-22.17
204	1231	0.00	0.00	351.00	1048.75
$\bar{X}$=12.75	$\bar{Y}$=76.94				

For example of TEST vs. QLTY,
For mean of QLTY ($\bar{Y}$),

$(\bar{Y}) = \frac{69+28+49+72+78+75+60+69+26+43+68+86+69+11+75+55}{16}$ = 58.31

$X - \bar{X}$ = 16- 12.75 = 3.25
$Y - \bar{Y}$ = 69 - 58.31 = 10.69
$(X - \bar{X})^2 = (16- 12.75)^2 = 10.56$
$(X - \bar{X})(Y - \bar{Y}) = 3.25 * 10.69 = 34.73$
Predicted value of Y for external code quality,

$b_1 = \frac{\Sigma(X-\bar{X})(Y-\bar{Y})}{\Sigma(X-\bar{X})^2} = \frac{825.25}{351.00} = 2.35$

$b_0 = \hat{y} - b_1\bar{x}$ = mean ($\bar{Y}$) -2.35 * mean ($\bar{X}$) = 58.31 – (2.35 * 12.75) = 28.34
$\hat{y} = b_0 + b_1x$ = 28.34+ (2.35* 12.75) =58.31
Predicted value of Y for developer productivity,

$b_1 = \frac{\Sigma(X-\bar{X})(Y-\bar{Y})}{\Sigma(X-\bar{X})^2} = \frac{1048.75}{351.00} = 2.99$

$b_0 = \hat{y} - b_1\bar{x}$ = mean ($\bar{Y}$) -2.99 * mean ($\bar{X}$) = 76.94 – (2.99 * 12.75) = 38.8 4
$\hat{y} = b_0 + b_1x$ = 38.84+ (2.99* 12.75) =76.94

5.4. Assessment

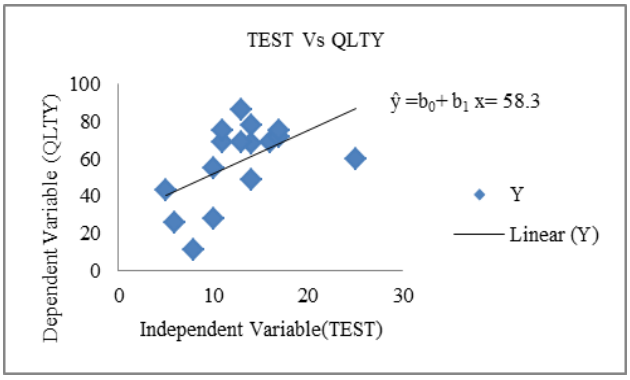

Fig. 4: QLTY on a function of TEST

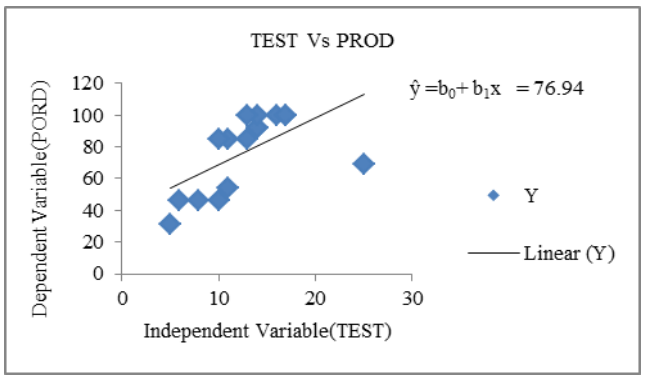

Fig. 5: PROD on a function of TEST

This analytical tool is most often applied to describe data motions over a period of time or correlation between two variables. This system expects that the regression analysis of the information compiled from the

developer productivity by TDD responds positively to questions RQ2. Due to the fact that TDD displays more steps in its process RQ1, a slight decrease in the external code quality is predicted.

In figure 4, the external code quality is slightly decreased by measuring the ordinary least squares (OLS) method of quality (QLTY) [12-13]. In figure 5, the developer productivity is improved by measuring the ordinary least squares (OLS) method of productivity (PROD).

6. Discussion and Comparison of Results

In this section, this paper presents the results acceptance test suite metric of regression analysis. Further, a significant relation between TEST and QLTY, as expressed in RQ1, with a positive was found. Hence, scatter plot figure 2 is an arithmetically expressive relationship between the number of tests and external code quality. Additional, a significant relation between TEST and PORD, as expressed in RQ2, with a somewhat down was found. So, scatterplot figure 3 is an arithmetically expressive correlation between the number of tests and programmer productivity. In this study, the number of tests is a good predictor for TDD programmer productivity [13]. Consequently, developer productivity over the number of tests becomes lightly diminishment and external code quality over the number of tests becomes improvement. Next, this paper presents the results of linear regression analysis. The predicted value of Y ($\hat{y}$) correlation between TEST and QLTY is 58.31. Further, a significant relation between TEST and QLTY, as expressed in RQ1, with a positive linear trend was not found. The linear regression between the two variables is expressed through the equation: QLTY= 28.34 + 2.35* TEST. This equation is plotted in Figure 2. The significance test for the linear regression coefficient, the regression line slope (b_1) is 2.35 and the regression intercept line of y-axis (b_0) is 28.34. Hence there is no arithmetically expressive relationship between the number of tests and external code quality. The predicted value of Y ($\hat{y}$) correlation between the TEST and PROD variables is 76.94. Further, a significant relation between TEST and PORD, as expressed in RQ2, with a positive linear trend was found. The linear regression between the two variables is expressed through the equation: PROD= 38.84 + 2.99* TEST. This equation is plotted in Figure 3. The significance test for the linear regression coefficient, the regression line slope (b_1) is 2.99 and the regression intercept line of y-axis (b_0) is 38.84. Hence there is an arithmetically expressive correlation between the number of tests and programmer productivity [13-14]. In this study, the number of tests is a good predictor for TDD programmer productivity. Consequently, developer productivity becomes improvement and external code quality becomes lightly diminishment.

7. Conclusion

The proposed system is a developing software technology that can support developers to design a code and in their task with resolution. Therefore, the developer will be capable to create extra reliable software. This system has given the developers a more logical accepting of their code and has supported them to advance their development skills. The system counts the bugs and defects over the time-frame. This approach allows thorough unit testing which enhances the quality of the software and advances customer satisfaction. They help with retaining and varying the code. Moreover, the number of acceptance test cases passed and number defects found through static code analysis are used to measure the external code quality. All these measures are consistent with the studies and will be considered as standard measures. When this proposed system assesses the acceptance test suite metric of regression analysis, the result of developer productivity over the number of tests is fewer decreased and the result of external code quality over the number of tests is increased in giving a fixed time-frame. The metric for external code quality and developer productivity used in this system clearly effect. If higher external code quality, lower developer productivity and vice versa, a developer may perhaps be more beneficial by executing as many user stories as possible but dismissing external code quality.

8. Acknowledgments

This research paper is partially supported by academic studies. Professionals were fit to implement more effective with test-driven development. Furthermore, this proposed system observes that the measurement reveal different aspects of a development approach in academic studies.

9. References

[1] Causineou and Chartier, 2010; Outliers Detection and Treatment: a Review, International Journal of Psychological Research, 3(1): 58-67.}

[2] H. Kou, P. M. Johnson, and H. Erdogmus, "Operational definition and automated inference of test-driven development with Zorro," Automated Software Engineering, 2010.

[3] Shaweta Kumar, Sanjeev bansal, "Comparative Study of Test driven Development with Traditional Techniques"; International Journal of Soft computing and Engineering (IJSCE); ISSN:2231-2307,Volume-3, Issue-1, (March 2013).

[4] A.N. Seshu Kumar and S. Vasavi ; "Effective Unit Testing Framework for Automation of Windows Applications"; Aswatha Kumar M.et al.(Eds); Proceedings of ICADC, AISC 174, pp. 813-822. Springerlink .com @ Springer India 2013

[5] Y. Rafique and V. B. Mi˘si´c, "The effects of test-driven development on external quality and productivity: A meta-analysis," IEEE Transactions on Software Engineering, vol. 39, no. 6, pp. 835–856, 2013.

[6] Causevic, A., Shukla, R., & Punnekkat, S. (2013). "Industrial study on test driven development: Challenges and experience" 2013 1st International Workshop on Conducting Empirical Studies in Industry (CESI).

[7] Davide Fucci, Burak Turhan, "On the role of tests in test- driven development: A differentiated and partial replication", Empirical Software Engineering Journal (April 2014, Volume 19, Issue 2, pp 277-302)

[8] Tosun A., Dieste O., Fucci D., Vegas S., Turhan B., Erdogmus H., Santos A., Oivo M., Toro K., Jarvinen J., & Juristo N. An Industry Experiment on the Effects of Test-Driven Development on External Quality and Productivity

[9] Fucci, D., Turhan, B., & Oivo, M. The Impact of Process Conformance on the Effects of Test-driven Development (ESEM2014) 8th Empirical Software Engineering and Measurement, 2014 ACM/IEEE International Symposium on. Turin, Italy.

[10] Fucci, D., Turhan, B., & Oivo, M. On the Effects of Programming and Testing Skills on External Quality and Productivity in a Test-driven Development Context (EASE2015) 19th Evaluation and Assessment in Software Engineering 2015 ACM/IEEE International Conference on., Nanjing, China.

[11] Viktor Farcic , Alex Garcia ; "Java Test-Driven Development"; First published: August 2015; Production reference: 1240815; Published by Packt Publishing Ltd.; Livery Place; 35 Livery Street; Birmingham B3 2PB, UK. ISBN 978-1-78398-742-9; www.packtpub.com; www.it-ebooks.in

[12] Christine Sarikas (GENERAL_EDUCATION) https:// blog. prepscholar.com/independent-and-dependent-variables; Feb 12, 2018.

[13] Tosun, A., Ahmed, M., Turhan, B., & Juristo, N. (2018). On the effectiveness of unit tests in test-driven development. Proceedings of the 2018 International Conference on Software and System Process - ICSSP '18.

[14] Fucci D., Scanniello G., Romano S., Shepperd M., Sigweni B., Uyaguari F., Turhan B., Juristo N., & Oivo M. "An External Replication on the Effects of Test-driven Development Using Blind Analysis" (ESEM2016) 10th Empirical Software Engineering and Measurement 2016 ACM/IEEE International Symposium on., Ciudad Real, Spain.