Proceedings of 10th International Conference on Science and Engineering 2019,
7-8 December 2019, Yangon, Myanmar

ICSE2019-ICT-6

Impact of Dependent and Independent Variables based on Ordinary Least Squares Method Using Test-Driven Development Approach

Myint Myint Moe[#1], Khine Khine Oo[*2]
[#] Faculty of Information Science department, University of Computer Studies (Hpa-an)
Kayin State, Myanmar
[*] Faculty of Information Science department, University of Computer Studies, Yangon
Myanmar
[1] myintmyintmoe.ucsy.1971@gmail.com
[2] khinekhineoo@ucsy.edu.mm

***Abstract*- Test-driven development (TDD) is a foundation for software evolution but unit tests must be performed before production code. To enhance both external code quality and programmers' productivity can be insisted on the exponents of TDD. The consequence of test-driven development on product quality and programmer productivity is analyzed the main purpose of this paper. This system builds the ordinary least squares method of regression analysis to assess the impact of the process on dependent variables and independent variables. This paper's results observed the positive effect of developer productivity, and slightly decrease the effect of external quality. TDD can affect advance software products' quality, also mend programmers' productivity. TDD undertook to help the delivery of high-quality products, both operational (fewer bugs) and technical perspective (cleaner code) while improving developers' productivity. TDD leads to less defects and fewer debugging period which correct code can be assured by writing tests first and thus serving the developer get a finer understanding of the software requirements. When this proposed system evaluates the ordinary least squares of regression analysis based on a fixed time-frame, the result of external code quality is fewer reduced, and the result of developer productivity is progressed.**

***Keywords*-** Test-Driven development, Number of tests, External Quality, Developer Productivity

I. INTRODUCTION

Test-driven development (TDD) is the basic segment of the agile code development approach by driving from Extreme Programming (XP) and the principles of the Agile Platform. In recent years, this approach has become popular in the industry as a requirements specification method. Before the code development, developers encourage to compose tests. 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. A unit test is an automated piece of code that applies a unit of work in the system and then a single notion about the action of that unit of work. 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 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. Before the actual development of the application, TDD is a process of developing and running automated tests.

TDD is intended to make the code clearer, simple and bug-free. This proposed system analyses 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). This decreases the fault of enhanced software either instantly or in the long run. The benefits of TDD, specifically improved software quality (external code quality) and speed up the testing process (developer productivity). This approach aims more productive and make fewer efforts per line of code. It perceives the attribute of the relationship between external code quality and developers' productivity. Number of tests, External code quality and developers' productivity focus on ordinary least squares method of regression analysis in statistics. 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.

The contribution of this paper observed the number of tests (#TESTS), 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.

This paper is organized as follows. Section (1) introduces the Test-Driven Development. A framework characterizing the whole procedure of test-driven development is presented in Section (2). Next, observational analysis of the proposed system is discussed in Section (3). Section (4) expresses discussion and comparison of results. Finally, Section (5) concludes this paper.

II. BACKGROUND THEORY

Test-Driven Development is a coding technique. TDD accelerates the early development of tests, at the time changes are received and encouraged with functional components. Test-Driven Development, invented by Kent Beck (inventor of Extreme Programming and JUnit) refers to a style of programming where three activities are closely intertwined: Coding, Testing (in the form of unit tests) and Design (in the form of refactoring). 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 user stories, are designed to simply express and understand. These can be easy to change by the end-user as they like during the project's handle time.

A. Test-Driven Development

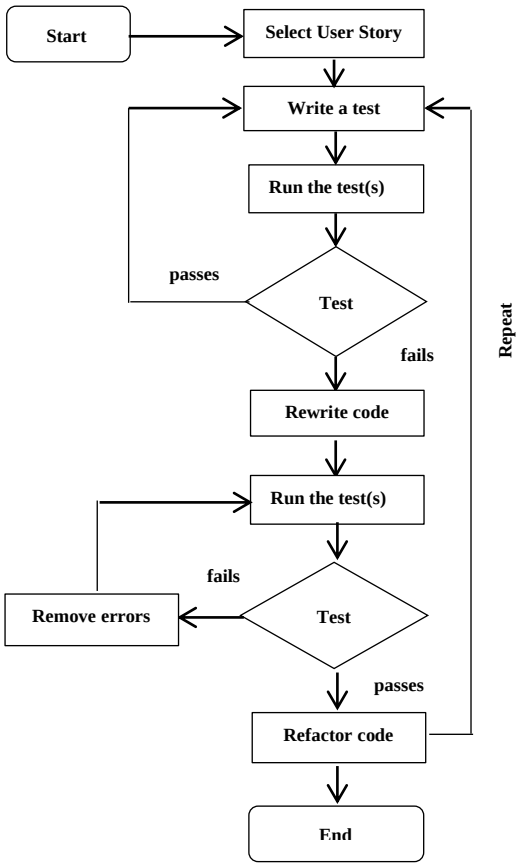

Figure 1: Test-Driven Development flow

The TDD process is presented in Figure 1, and consists of the following steps:

(1) Select a user story,
(2) Write a test that completes a little function of the user story and that delivers a missed test,
(3) Re-write the creation code necessary to implement the feature,
(4) Execute the preceding tests again. Where if any test fails, the code is corrected and the test set is re-executed and finally,
(5) Production code and the tests are refactored.

As the refactoring stage is finished, the user can select the new user story again. This method produces some benefits that focus on the promise of increasing the quality of the software product and the productivity of programmers.

III. PROPOSED SYSTEM

In this proposed method, the linear regression analysis uses to measure the number of tests, external code quality and developer productivity.

A. Research Questions

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

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 ordinary least squares of regression analysis in statistics.

B. Method

QLTY and PROD are the dependent variables. #TESTS is the independent variable. The data set consisting of #TESTS, QLTY and PROD attributes were analyzed to discover outliers using both z-score and modified z-score methods [1, 7]. 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. #TESTS assessed by the count of the JUnit test cases. This is a ratio variable within the range [0, ∞]. QLTY defined 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 represents the relationship between a known independent variable and an unknown dependent variable. In regression analysis, dependent variables are illustrated on the vertical y-axis, while independent variables are illustrated on the horizontal x-axis. The line of best-fit decided from the least-squares method has an equation that states the 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 change for each 1 unit change in the X variable. The b_0 is the intercept of the regression line with the y-axis. In other words, it is the value of Y if the value of X = 0. Y-hat = $b_0 + b_1(x)$ is the sample regression line. This paper must assess b_0 and b_1 to construct 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:

$$\text{Mean of \#TEST } (\bar{X}) = \frac{\Sigma X}{N} \quad (1)$$

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

$$\text{Mean of QLTY (or) PROD } (\bar{Y}) = \frac{\Sigma Y}{N} \quad (2)$$

ΣY = sum of all the individual QLTY or PROD data set

N = total number of QLTY (or) PROD data set

Predicted value of Y,

$b_0 = \hat{y} - b_1 x = \text{mean}(\bar{Y}) - b_1 * \text{mean}(\bar{X})$ (3)

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

$$b_1 = \frac{\Sigma(X - \bar{X})(Y - \bar{Y})}{\Sigma(X - \bar{X})^2} \quad (4)$$

b_1= the slope of the regression line

$$\hat{y} = b_0 + b_1 x \quad (5)$$

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

TABLE 1
DATASET USED IN THE ASSESSMENT

#TESTS	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

The table-1 dataset consisting of #TESTS, QLTY and PROD attributes was tested to find outliers using both z-score and modified z-score method**s** [1]**.** This paper used the dataset from Davide Fucci and Burak Turhan (April 2014) [7].

For example of #TESTS 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,

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

$b_0 = \hat{y} - b_1 x = \text{mean}(\bar{Y}) - 2.35 * \text{mean}(\bar{X})$

$= 58.31 - (2.35 * 12.75)$

$= 28.34$

$\hat{y} = b_0 + b_1 x = 28.34 + (2.35 * 12.75) = 58.31$

Predicted value of Y for developer productivity,

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

$b_0 = \hat{y} - b_1 x = \text{mean}(\bar{Y}) - 2.99 * \text{mean}(\bar{X})$

$= 76.94 - (2.99 * 12.75)$

$= 38.8\ 4$

$\hat{y} = b_0 + b_1 x = 38.84 + (2.99 * 12.75) = 76.94$

TABLE 2
DATA OF COMPUTATION FOR CORRELATION OF #TESTS 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 3
DATA OF COMPUTATION FOR CORRELATION OF #TESTS AND PORD

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**				

C. 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. A trend-line also referred to as a line of best fit, is a straight or curved line in a chart that shows the general pattern or overall direction of the data. This analytical tool is most often used to show data movements 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*. A slight decrease in the external code quality is expected due to the fact that TDD presents more steps in its process *RQ1*.

In figure 2, the external code quality is slightly decreased by measuring the ordinary least squares (OLS) method of quality (QLTY).

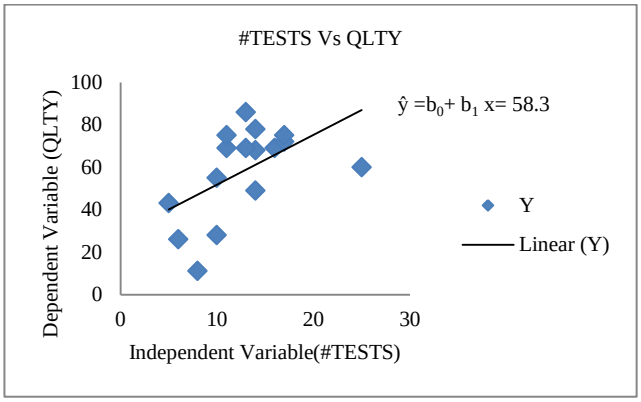

Figure 2 QLTY as a function of #TESTS

In figure 3, the developer productivity is improved by measuring the ordinary least squares (OLS) method of productivity (PROD).

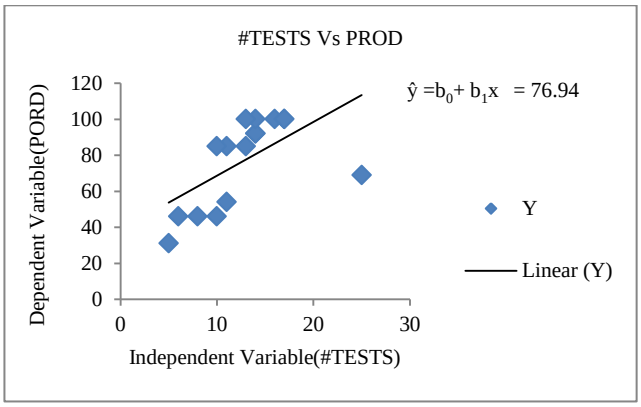

Fig. 3 PROD as a function of #TESTS

IV. DISCUSSION AND COMPARISON OF RESULT

In this section, this paper presents the results of linear regression analysis. The predicted value of Y (ŷ) correlation between #TESTS and QLTY is 58.31. Further, a significant relation between #TESTS 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* #TESTS. 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 (ŷ) correlation between the #TESTS and PROD variables is 76.94. Further, a significant relation between #TESTS 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* #TESTS. 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. 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.

V. CONCLUSIONS

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 maintaining and changing 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 ordinary least squares of regression analysis, the result of external code quality is fewer decreased, and the result of developer productivity is increased in giving a fixed time-frame.

ACKNOWLEDGMENT

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.

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.info.

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

[13] https://chartio.com/learn/charts/what-is-a-scatter-plot/__Jan 9, 2019

[14] Svetlana Cheusheva ; https://www.ablebits.com/office-addins-blog/2019/01/09/add-trendline-excel/ May 15, 2019

[15] Will Kenton_https://www.investopedia.com/terms/l/least-squares-method.asp; Sep 2, 2019.

[16] 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.