Scientific Journal of Informatics
Vol. 7, No. 2, November 2020

p-ISSN 2407-7658 http://journal.unnes.ac.id/nju/index.php/sji e-ISSN 2460-0040

Test-Driven Development (TDD) for Point of Sale System at Bicycle Shop

Egia Rosi Subhiyakto[1], Yani Parti Astuti[2]

[1,2]Informatics Engineering Department, Faculty of Computer Sciences,
Universitas Dian Nuswantoro Semarang, Indonesia
Email: [1]egia@dsn.dinus.ac.id, [2]yani.parti.astuti@dsn.dinus.ac.id

Abstract

The information technology systems are developing faster; one of its developments is a system to process the recording of sales data (Selling System). To support the selling process in Ali Cycle bicycle shop, which previously done manually, it needs a system to record the stock of goods, transaction, supplier, and sales report. There is a lot of model in building the system, one of the model is traditional development model, generally, the phase of this model is never-ending related to system problems or bugs, sometimes bugs are not found in the development but comes after practical use begins. A Point of Sale (POS) system with Test Driven Development (TDD) method has been built in which a test is written before the coding phase in a purpose of the codes, which are created, has passed the test, reducing the bug and it tests the system. The results show that all codes have passed the test; the test consists of 89 functions and 397 statements. Evaluation results of end-users testing showed that the majority of respondents strongly agree and agree with a system with an average rating of 94% for performance, 89% interface and 83% user satisfaction. The conclusion is, building a POS system using TDD method succeeded by producing a useful system as the requirement and expected system quality.

Keywords: Point of Sale, TDD, Testing, Quality

1. INTRODUCTION

The development of information technology is faster along with the ability of computers in the data processing. One of the developments is the information system for the process of recording company sales data or often referred to as Point of Sale (POS). POS system can be said to be good where the system has these features including, can see data categories of goods, see data on goods, stock of goods, sales reports, and sales transaction data. This sales information system is still much needed by the community to manage and facilitate their sales efforts, one of which is Ali Cycle. Ali Cycle is a business that focuses on service and sales services for various types of bicycles ranging from mountain bikes, road bikes, BMX bikes, folding bikes and so on. Meanwhile, to manage this business, the owner must present information on sales transactions, stock information, and sales reports. All that is still conventional, the more years the business is growing, which means that sales transaction information, stock items, and sales reports are increasing and very ineffective if done manually [1]. Indeed all of that can be done manually but it is very ineffective and takes a lot of time too. Therefore, software development is needed, namely POS, which can manage the business effectively.

In building software, many development models have been developed over the years with varying degrees of success. These include the Waterfall Model, Iterative Development, Prototyping, Spiral Model, RAD or often referred to as the traditional development model [2,3]. This traditional model is rarely ending, evolutionary of software development described in [3]. Generally, system problems or bugs that are developed (which are not found during the development life cycle) appear after practical use begins, so that problems related to the system are solved after system implementation. Also in this traditional development model, all requirements must be known at the beginning or in advance of development, some of the same code or redundant that has been implemented has never been run by the system at all. All these shortcomings affect the performance and speed of the system is built[2]. These deficiencies sometimes make IT professionals in the industrial environment from IT architects to programmers unable to explain the advantages and disadvantages obtained[4]. Of the shortcomings of the traditional development model, the Test Driven Development (TDD) method is the right solution. TDD promised increased quality and productivity [5]. With the TDD method to make the system made more testable or tested, thus reducing the possibility of system errors when developing or when it has been deployed [2]. In [6] combines several techniques including TDD for an agile formal development process. Some of the studies using TDD include [7], [8], [9], [10], [11], [12], [13], [14]. Meanwhile [15] using TDD to selecting UML models along development process. The advantages of test-driven development described in [16].

This study developed a Point of a Sale information system at the Ali Cycle bicycle shop using the Test Driven Development method based on the Web App. To produce a system that has many features to facilitate sales management including product management features, suppliers, sales transactions, and inventory that can run well and quickly than there are many defects or bugs.

2. METHODS

Research conducted using one of the many practices of Extreme Programming development methods, namely Test-Driven Development (TDD) or often also known as Unit Test-first Development, namely by making test cases before creating functional code. In the [17] propose an additional benefit of Acceptance TDD and [18] adding criteria to TDD. On the other hand, [19] investigate how the TDD characteristics impact quality and productivity. Then, [20] experimenting with the effect of TDD on productivity, code, and tests. More complete, [21] conducts a review on the effect of TDD on internal quality, external quality, and productivity. In the [27] conducted an experiment with 24 professionals from three different sites of a software organization to implemented TDD and ITLD. It was observed that the subjects were more productive when applying TDD to simple tasks compared to ITLDs. In the other hand, the effectiveness of TDD has been proven in [25], [26]. Figure 1 shows the 5 steps or phases in applying this method [2].

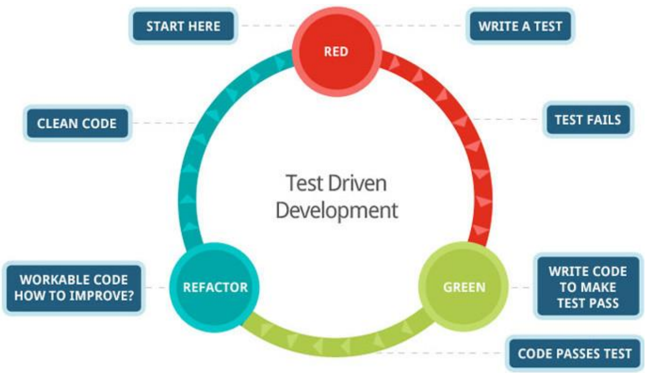


Figure 1. The flow of software development with TDD [2]

2.1. Write a test

The first step is to make a unit test of each feature that will be created. In this study, unit tests that will be made include starting from product management features, suppliers, sales transactions, and inventory. When there are new features, it starts with writing a test. Making your scenario test requires a clear description of the features, specifications until the needs begin from input to output to be issued.

2.2. Run a test

When you have made a test of each feature starting from the features of sales transactions, inventory, and others, make sure when running a test of features that have been made the results fail or do not meet the expected results. At this stage, it is often referred to as red-phase, because the functional code has not been created.

2.3. Writing functional code

At this stage, writing program code, program code that will be made in this study includes the features of sales transactions and inventory. The main goal is to make the program code that is made to pass the test. In this section, the program code for sales, inventory, or other transaction features can still be messy or unreadable, because this stage has the main goal of passing the test and not having to make additional functions other than to pass the test.

2.4. Re-run the test

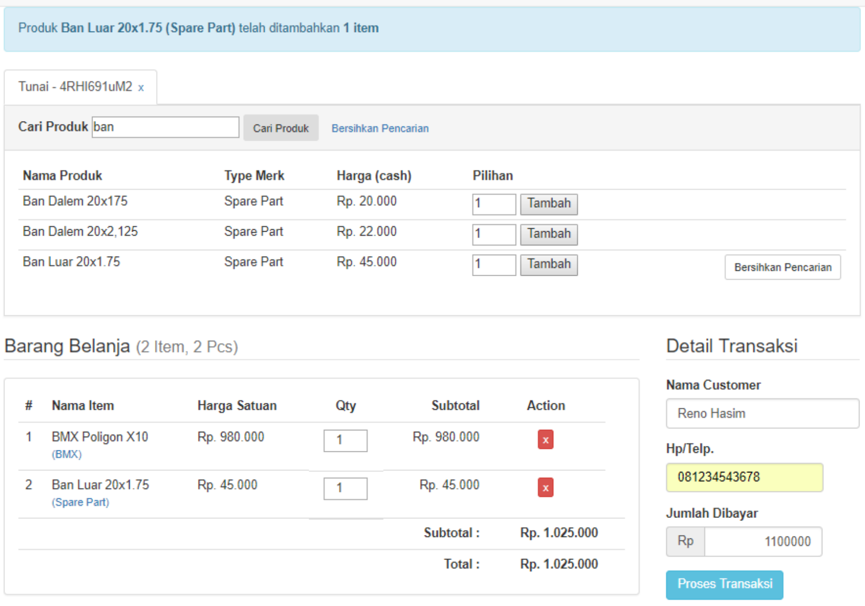
At this stage make sure the program code that is created starting from the features of sales transactions, inventory and others have passed the test, if not then do iterations on making the program code until all have passed the test. With continuous iteration, the programmer can easily monitor system performance while avoiding errors or bugs in the program code after changes or adding new features.

2.5. Refactoring

It is at this stage that the program code which is still cluttered, duplicate and inefficient can be tidied up, with this testing iteration while ensuring the changes made produce something similar before refactoring. With the development of code applications that are made more and more, the TDD approach can help the code that is made easier to read again and easily maintained by the programmer, or other programmers.

3. RESULT AND DISCUSSION

This section discusses the results of system implementation and testing that have been carried out. Implementation is carried out using the PHP programming language with the Laravel framework from the Back-End side, for making unit tests using the PHPUnit package and from the Front-End side using HTML, CSS, and JavaScript. Following are the results of TDD implementation according to the stages in one of the features of the Point of Sale (POS), namely the Transaction feature. Figure 2 shows when the user wants to make a new transaction, the first step is to find the product the customer wants to buy on the search form and when the product is selected it will automatically enter the shopping basket, then fill in the customer's name, telephone number, and the amount of money get paid. In the other hand, the steps to implement TDD on the transaction feature can be seen in Figure 2


Produk **Ban Luar 20x1.75 (Spare Part)** telah ditambahkan **1 item**

Tunai - 4RHI691uM2 x

Cari Produk ban

Nama Produk	Type Merk	Harga (cash)	Pilihan
Ban Dalem 20x175	Spare Part	Rp. 20.000	1 Tambah
Ban Dalem 20x2,125	Spare Part	Rp. 22.000	1 Tambah
Ban Luar 20x1.75	Spare Part	Rp. 45.000	1 Tambah

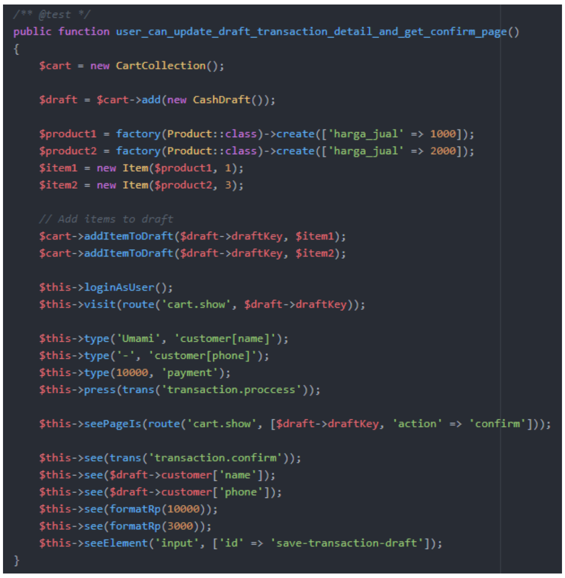
Barang Belanja (2 Item, 2 Pcs)

#	Nama Item	Harga Satuan	Qty	Subtotal	Action
1	BMX Poligon X10 (BMX)	Rp. 980.000	1	Rp. 980.000	x
2	Ban Luar 20x1.75 (Spare Part)	Rp. 45.000	1	Rp. 45.000	x
				Subtotal :	Rp. 1.025.000
				Total :	Rp. 1.025.000

Detail Transaksi

Nama Customer	Reno Hasim
Hp/Telp.	081234543678
Jumlah Dibayar	Rp 1100000

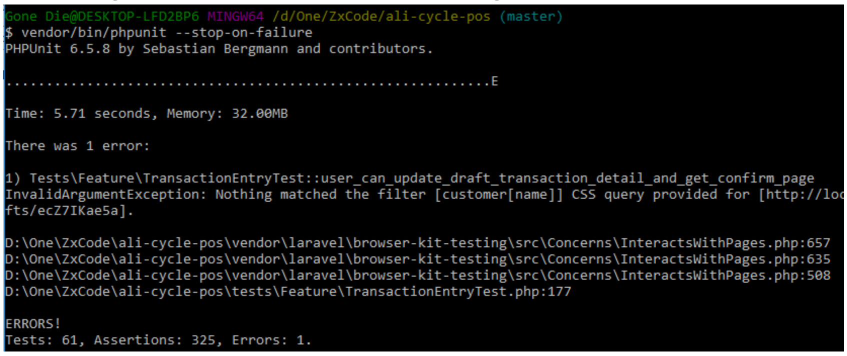
Figure 2. Results of transaction feature



```php
/** @test */
public function user_can_update_draft_transaction_detail_and_get_confirm_page()
{
    $cart = new CartCollection();

    $draft = $cart->add(new CashDraft());

    $product1 = factory(Product::class)->create(['harga_jual' => 1000]);
    $product2 = factory(Product::class)->create(['harga_jual' => 2000]);
    $item1 = new Item($product1, 1);
    $item2 = new Item($product2, 3);

    // Add items to draft
    $cart->addItemToDraft($draft->draftKey, $item1);
    $cart->addItemToDraft($draft->draftKey, $item2);

    $this->loginAsUser();
    $this->visit(route('cart.show', $draft->draftKey));

    $this->type('Umami', 'customer[name]');
    $this->type('-', 'customer[phone]');
    $this->type(10000, 'payment');
    $this->press(trans('transaction.proccess'));

    $this->seePageIs(route('cart.show', [$draft->draftKey, 'action' => 'confirm']));

    $this->see(trans('transaction.confirm'));
    $this->see($draft->customer['name']);
    $this->see($draft->customer['phone']);
    $this->see(formatRp(10000));
    $this->see(formatRp(3000));
    $this->seeElement('input', ['id' => 'save-transaction-draft']);
}
```

Figure 3. Implementation of TDD on transaction features

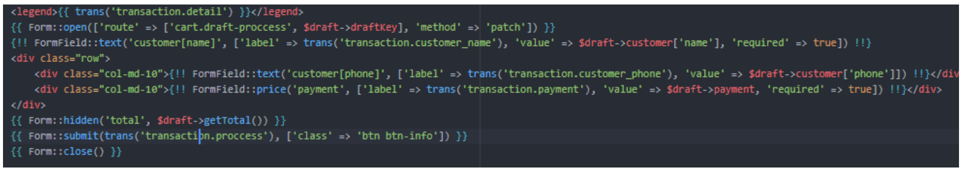
The first step in implementing TDD is to make a test like Figure 3. And proceed with running the test that has been made like Figure 4.



```
Gone Die@DESKTOP-LFD2BP6 MINGW64 /d/One/ZxCode/ali-cycle-pos (master)
$ vendor/bin/phpunit --stop-on-failure
PHPUnit 6.5.8 by Sebastian Bergmann and contributors.

............................................................E

Time: 5.71 seconds, Memory: 32.00MB

There was 1 error:

1) Tests\Feature\TransactionEntryTest::user_can_update_draft_transaction_detail_and_get_confirm_page
InvalidArgumentException: Nothing matched the filter [customer[name]] CSS query provided for [http://loc
fts/ecZ7IKae5a].

D:\One\ZxCode\ali-cycle-pos\vendor\laravel\browser-kit-testing\src\Concerns\InteractsWithPages.php:657
D:\One\ZxCode\ali-cycle-pos\vendor\laravel\browser-kit-testing\src\Concerns\InteractsWithPages.php:635
D:\One\ZxCode\ali-cycle-pos\vendor\laravel\browser-kit-testing\src\Concerns\InteractsWithPages.php:508
D:\One\ZxCode\ali-cycle-pos\tests\Feature\TransactionEntryTest.php:177

ERRORS!
Tests: 61, Assertions: 325, Errors: 1.
```

Figure 4. Running test results

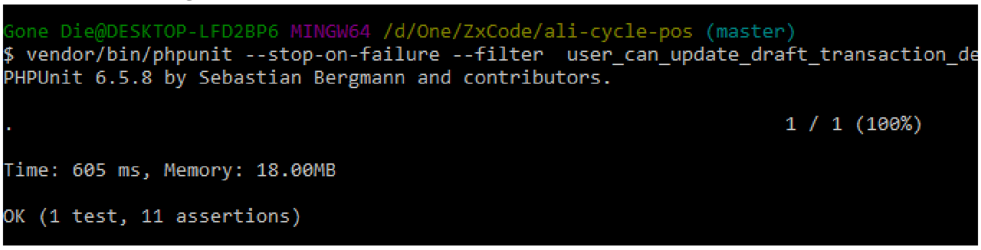
The test results above show Errors because the test that was made does not see the form to enter data in the form of the customer's name. Then the functional code or process step 3 is made in the TDD method to meet the errors above.



```
<legend>{{ trans('transaction.detail') }}</legend>
{{ Form::open(['route' => ['cart.draft-proccess', $draft->draftKey], 'method' => 'patch']) }}
{!! FormField::text('customer[name]', ['label' => trans('transaction.customer_name'), 'value' => $draft->customer['name'], 'required' => true]) !!}
<div class="row">
    <div class="col-md-10">{!! FormField::text('customer[phone]', ['label' => trans('transaction.customer_phone'), 'value' => $draft->customer['phone']]) !!}</div
    <div class="col-md-10">{!! FormField::price('payment', ['label' => trans('transaction.payment'), 'value' => $draft->payment, 'required' => true]) !!}</div>
</div>
{{ Form::hidden('total', $draft->getTotal()) }}
{{ Form::submit(trans('transaction.proccess'), ['class' => 'btn btn-info']) }}
{{ Form::close() }}
```

Figure 5. Writing user functional codes filling transaction details

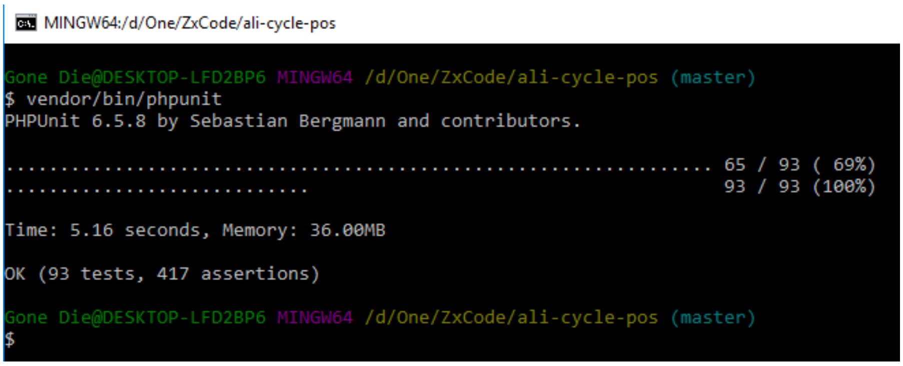
Figure 5 shows the functional code for the customer's name form, the customer's telephone number, and the amount of money paid by the customer. The next step is to re-run the test and see the results whether it meets the test or not. The results can be seen in Figure 6.



```
Gone Die@DESKTOP-LFD2BP6 MINGW64 /d/One/ZxCode/ali-cycle-pos (master)
$ vendor/bin/phpunit --stop-on-failure --filter  user_can_update_draft_transaction_de
PHPUnit 6.5.8 by Sebastian Bergmann and contributors.

.                                                                  1 / 1 (100%)

Time: 605 ms, Memory: 18.00MB

OK (1 test, 11 assertions)
```

Figure 6. Results of running a user test filling transaction details

When you have made the code in the form with the customer's name, customer telephone number, and the amount paid by the customer, then try again to run the test, and the results turned out to have passed the test or OK, as in Figure 6. So for the user feature can fill in transaction details that have been tested because it has passed the test. While testing for all features on this POS system the results are OK or all the codes of all features have passed we can see the test results in Figure 7.



```
MINGW64:/d/One/ZxCode/ali-cycle-pos

Gone Die@DESKTOP-LFD2BP6 MINGW64 /d/One/ZxCode/ali-cycle-pos (master)
$ vendor/bin/phpunit
PHPUnit 6.5.8 by Sebastian Bergmann and contributors.

.................................................................  65 / 93 ( 69%)
............................                                       93 / 93 (100%)

Time: 5.16 seconds, Memory: 36.00MB

OK (93 tests, 417 assertions)

Gone Die@DESKTOP-LFD2BP6 MINGW64 /d/One/ZxCode/ali-cycle-pos (master)
$
```

Figure 7. Results of running tests on all features

After testing in the application development environment done, then testing will go at the end-user or end-user level. This section uses MOS (Mean Opinion Score) to present the evaluation results got by testing end-users regarding their perceptions of this application. We use evaluation to measure system performance, display form, and user satisfaction, as done by [22]. The quality of testing in TDD is described in [23]. System performance parameters and display form affect user satisfaction in using the system built. Stages of testing do not guarantee the quality of the system but can provide a level of confidence in the system. Software testing techniques described in [24]. Users who took part in this study were 15 respondents.

We give each respondent a brief explanation of the system being built. Then the respondent is given time to use the system. There were eleven questions given, which included the speed of login access, not finding bugs, speed of access to all features, the system could run according to initial requirements, the display was interactive; the display was proportional, the display system was interesting, user satisfaction, until giving recommendations to friends respondents to use the system. Based on the evaluation results of the questionnaire given will analyzed using a Likert scale of 1 to 5, strongly agree, agree, neutral, disagree, and strongly disagree. The percentage is got by the formula $Y = P / Q \times 100\%$, where Y is the percentage value, P is the number of respondents who answered the question, and Q is the number of respondents. Evaluation results are divided into three main parts, namely performance evaluation, display form evaluation, and satisfaction evaluation. Evaluation results of end-users testing showed that the majority of respondents strongly agreed and agreed with a system with an average rating of 94% for performance, 89% interface and 83% user satisfaction.

4. CONCLUSION

The sales system or Point of Sale (POS) for Ali Cycle bicycle shop makes it very easy to manage its. Besides this system was also built using the Test Driven Development (TDD) development method to reduce the possibility of errors or bugs appearing on the system during development or when it has been deployed. It can run the results of testing the functional code of the system using the TDD method on all test functions in which there are test scenarios of each feature, and the result is that all code that has been made all have passed the test. The test results comprise 93 test functions and 417 statements or input from the test data that has been made.

5. REFERENCES

[1] Kaur, R., & Sengupta, J. (2013). Software process models and analysis on failure of software development projects. *arXiv preprint arXiv:1306.1068*.

[2] Kumar, S., & Bansal, S. (2013). Comparative study of test driven development with traditional techniques *International Journal of Soft Computing and Engineering (IJSCE), 3*(1), 2231-2307.

[3] K. Bajaj, H. Patel, and J. Patel, (2015). Evolutionary software development using Test Driven approach, *2015 International Conference and Workshop on Computing and Communication. IEMCON 2015*.

[4] Latorre, R. (2014). A successful application of a Test-Driven Development strategy in the industrial environment. *Empirical Software Engineering*, *19*(3), 753-773.

[5] Karac, I., & Turhan, B. (2018). What do we (really) know about test-driven development?. *IEEE Software*, *35*(4), 81-85.

[6] Aichernig, B. K., Lorber, F., & Tiran, S. (2014, January). Formal test-driven development with verified test cases. In *2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)* (pp. 626-635). IEEE.

[7] Bouraqadi, N., & Mason, D. (2018). Test-driven development for generated portable Javascript apps. *Science of Computer Programming*, *161*, 2-17.

[8] Besson, F., Moura, P., Kon, F., & Milojicic, D. (2015). Bringing Test-Driven Development to web service choreographies. *Journal of Systems and Software*, *99*, 135-154.

[9] Borle, N. C., Feghhi, M., Stroulia, E., Greiner, R., & Hindle, A. (2018). Analyzing the effects of test driven development in GitHub. *Empirical Software Engineering*, *23*(4), 1931-1958..

[10] Guduvan, A. R., Waeselynck, H., Wiels, V., Durrieu, G., Fusero, Y., & Schieber, M. (2013, June). STELAE—A model-driven test development environment for avionics systems. In *16th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2013)* (pp. 1-8). IEEE.

[11] J. W. Burris, (2017). Test-driven development for parallel applications, *Proc. - 2017 2nd Int. Conf. Inf. Syst. Eng. ICISE 2017*, vol. 2017-January, 27–31.

[12] Hamill, P., Alexander, D., & Shasharina, S. (2009, July). Web service validation enabling test-driven development of service-oriented applications. In *2009 Congress on Services-I* (pp. 467-470). IEEE.

[13] Laranjeiro, N., & Vieira, M. (2009, June). Extending test-driven development for robust web services. In *2009 Second International Conference on Dependability* (pp. 122-127). IEEE.

[14] Nanthaamornphong, A., & Carver, J. C. (2018). Test-Driven Development in HPC Science: A Case Study. *Computing in Science & Engineering*, *20*(5), 98-113.

[15] Hametner, R., Winkler, D., Östreicher, T., Surnic, N., & Biffl, S. (2010, September). Selecting UML models for test-driven development along the

automation systems engineering process. In *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)* (pp. 1-4). IEEE..

[16] Lenka, R. K., Kumar, S., & Mamgain, S. (2018, October). Behavior Driven Development: Tools and Challenges. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)* (pp. 1032-1037). IEEE.

[17] Fontela, C., & Garrido, A. (2013). Connection between safe refactorings and acceptance test driven development. *IEEE Latin America Transactions*, *11*(5), 1238-1244.

[18] Shelton, W., Li, N., Ammann, P., & Offutt, J. (2012, April). Adding criteria-based tests to test driven development. In *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation* (pp. 878-886). IEEE.

[19] D. Fucci, H. Erdogmus, B. Turhan, M. Oivo, and N. Juristo, (2017). A Dissection of the Test-Driven Development Process: Does It Really Matter to Test-First or to Test-Last?, *IEEE Trans. Softw. Eng,* 43, no. 7, . 597–614.

[20] Pančur, M., & Ciglarič, M. (2011). Impact of test-driven development on productivity, code and tests: A controlled experiment. *Information and Software Technology*, *53*(6), 557-573.

[21] Bissi, W., Neto, A. G. S. S., & Emer, M. C. F. P. (2016). The effects of test driven development on internal quality, external quality and productivity: A systematic review. *Information and Software Technology*, *74*, 45-54.

[22] Subhiyakto, E. R., & Utomo, D. W. (2017). RMTool; Sebuah Aplikasi Pemodelan Persyaratan Perangkat Lunak menggunakan UML. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI)*, *6*(3), 268-274.

[23] A. Čaušević, S. Punnekkat, and D. Sundmark, (2012). Quality of testing in test driven development, *Proceeding - 2012 8th International Conference on the Quality of Information and Communications Technology. QUATIC 2012,* . 266–271.

[24] Subhiyakto, E. R., & Utomo, D. W. (2016). Software Testing Techniques and Strategies Use in Novice Software Teams. *SISFO Vol 5 No 5*, *5*.

[25] Dogša, T., & Batič, D. (2011). The effectiveness of test-driven development: An industrial case study. *Software Quality Journal*, *19*(4), 643–661.

[26] Nanthaamornphong, A., & Carver, J. C. (2017). Test-Driven Development in scientific software: a survey. *Software Quality Journal*, *25*(2), 343-372.

[27] Tosun, A., Dieste, O., Fucci, D., Vegas, S., Turhan, B., Erdogmus, H., ... & Juristo, N. (2017). An industry experiment on the effects of test-driven development on external quality and productivity. *Empirical Software Engineering*, *22*(6), 2763-2805.