International Journal of Applied Science and Engineering

Hybrid model for evaluation of quality aware DevOps

Pooja Batra*, **Aman Jatain**

Department of Computer Science, Amity University, Haryana, India

ABSTRACT

Today most of the organizations are switching to DevOps for faster and reliable delivery. It promotes the collaboration between developers and IT teams. In ancient development practices, development and operation teams were working in silos and when DevOps introduced, it combines both teams to integrate and automate the processes. A single team with cross functioning members provides not only technical advantages but also cultural benefits. Faster delivery of software, less complex designs, stable operating environments, customer satisfaction is some of the outcomes of DevOps. Although it proved to be a responsive environment for software delivery yet lacks in quantifiable perspective. There is no metric defined to measure performance and that can be estimated by using key attributes of software. In this research hybrid framework is proposed to improve the software reliability and productivity. Proposed framework for DevOps is named TDMBD (Test Driven Measurement Based DevOps) which provides solutions to challenges in DevOps like performance issues, poorly defined methodologies, and unstandardized processes. Paper focuses on defining and measuring metrics that are derived from measurement-based system of software and TDMBD is evaluated based on metrics analysis. To validate results of proposed approach a comparison is shown in between existing approach and proposed approach. Finally, through proposed method better quality of product is retrieved.

Keywords: DevOps, Development and operations, Agile, CI/CD, Software development, Framework.

OPEN ACCESS

Received: May 24, 2021
Revised: June 23, 2021
Accepted: July 5, 2021

Corresponding Author:
Pooja Batra
poojabatra9@gmail.com

Publisher:
Chaoyang University of Technology
ISSN: 1727-2394 (Print)
ISSN: 1727-7841 (Online)

1. INTRODUCTION

DevOps is a collection of practices that is supposed to deliver faster and reliable software. This speed and reliability supports software industry to perform effectively in competitive market (Erich et al., 2014). DevOps is a combination of two processes: development and operation tasks. Two different teams are collaborated to work across the lifecycle of project from planning, development and test to deployment and operations (Hüttermann, 2012). DevOps has gone through a long journey from waterfall to agile approach. To manage deployment environment and configure automation, agile principles are used (Huttermann, 2012). Different departments have different goal that may leads to in collaboration and inefficiency. DevOps resolves these issues by introducing cross functional teams that are collaborative too. These teams' takes the responsibility of development process from scratch to reliable product delivery ensuring the quality in automation process also. Fig. 1 presents the generalized framework of DevOps.

Every organisation is taking benefits from DevOps process but cannot neglect the challenges in adopting and implementing DevOps. Software industry has its own purpose of using DevOps as it provides so many benefits but the overall motive is to get quality software. So it is necessary to clarify the purpose of DevOps implementation (Elberzhager et al., 2017; Floris et al., 2014). Performance can be measured through metrics calculation. So far developed frameworks of DevOps fulfilling the needs of

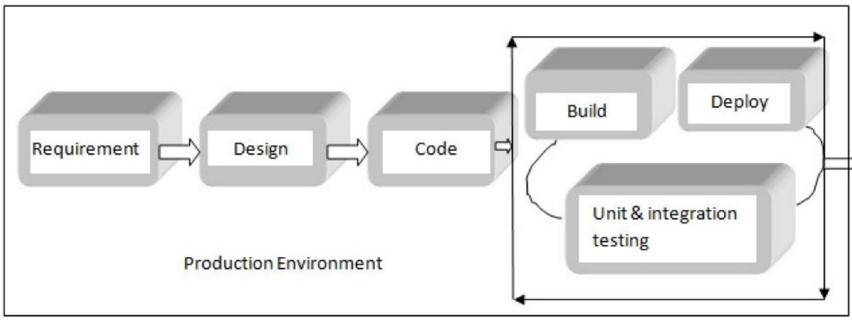

Fig. 1. Generalized framework of DevOps

organisation conceptually. There is no method to measure the success of these frameworks. For this purpose, metrics should be defined and evaluated which will be justified by a framework along with all drivers. The proposed research work is an effort to resolve these concerns and in this context. Test driven Measurement Based DevOps framework is proposed which is also an integrated framework (TDMBD) for assessment of quality of service (Qos). Existing frameworks discussed in literature mainly fiction based qualitative studies and do not support metrics evaluation. In the discussed methodology, the proposed framework is validated using quantitative measures and new metrics are defined to analyze and evaluate the framework.

Research paper is organized as: Section 1- introduces DevOps process along with the need of the framework. Section 2- describes the related study performed in the field of DevOps along with challenges. Evolution of test-driven approach is discussed here. In next section, metrics are defined and discusses the proposed framework and validating it by evaluation quality metrics. Last section consists of the conclusion remarks and future aspects of the research work.

2. RELATED WORK

DevOps evolves from agile methodology. The term DevOps is a combination of two terms that is Dev and Ops. Dev stands for development while Ops stands for operations. As it organization was looking for approach that is better than traditional approaches and can bridge the gap between development and operations departments. So DevOps is the system which fills the communicational and technological gap between developer and operations. It automats the processes like continuous development (CD), continuous testing and continuous integration (CI) (Shahin et al., 2017). In 2012 Michael Huttermann has given the concepts of DevOps in his book title “DevOps for developers” (Hüttermann, 2012). In the literature work DevOps process and underlying concepts were described in structural manner. Software delivery process is streamlined by the activities defined in DevOps and those activities starts from collecting requirement to taking feedback from customer after delivery. Nicolau de França et al. (2016) characterised DevOps. DevOps was characterized while collecting multiple views from every type of literature. Many benefits and challenges of DevOps were discussed in their work. DevOps not only provides faster delivery of software but also provide better quality.

DevOps is suffering from many challenges like no proper defined methodology or framework, no procedure to evaluate quality, performance issues and many more. According to Floris et al. (2014), DevOps was suffering from poor and low quality research. No process or methodology was not defined for DevOps. A conceptual framework was introduced to overcome this challenge. In Gotteshiem (2015) discussed performance issues in DevOps in his research work. According to research work DevOps lacks for performance metrics which are necessary to resolve performance issues. Various challenges related to DevOps were discussed in detail by Liu and Zhou (2017). Authors elaborated many issues of DevOps in detail. Quality standards are not defined properly in DevOps, quality metrics are not defined anywhere to validate quality process. There is need of effective risk management in DevOps approaches. As there are no quality standards so performance cannot be measured. Everyday practice of every software evolving process is to enhance quality and performance (Batra and Jatain, 2020a).

The proposed framework combines test driven and DevOps approach. Beck et al. (2001) has introduced the concept of test driven development in software development life cycle (Beck, 2001) Test-driven development is an evolutionary approach which provides test-first development where a test case is written before writing a code to fulfill that test case and after refactoring process occurs. Test driven development techniques has several benefits like reduced development time, increased productivity and many more (Madeyski and Szała, 2007).

Mäkinen and Münch (2014) found the impact of test driven development (TDD) on software quality. As per author’s research test driven development has less number of defects as compared to traditional software development. Even code was more simper, smaller and less complex and easy to maintain. These benefits of TDD approach and addressed issues in DevOps drove us to research on present topic and we came up with new framework which is validated through various performance metrics and resolving performance and other issues too.

3. PROPOSED METHODOLOGY

In this paper three layered architecture is presented to extend the DevOps process by applying various quality observations. As shown in Fig. 2 the proposed framework has combined the three major aspects of DevOps process at three different layers. Uppermost layer consists of approaches used in our DevOps framework. First approach is test driven which yields less number of defects and tighter collaboration of team members. It also performs well in the ever changing environment. Next approach is agile based approach resulting continuous and faster delivery of product (Nagarajan and Overbeek, 2018). Last used approach refers to continuous integration and continuous delivery. It not only supports automation process but results into quality product with faster delivery. Second layer refers to practices involved in proposed framework of DevOps process. Initiating from planning, test driven environment takes charge by writing executing test cases first and further build is prepared. Build is automatically tested and deployed. Operations procedures further takes ahead. Third layer specifies the production environment performance in terms of defect density, reliability, risk coverage, productivity and deployment frequency. A perfect combination of approaches and practices will yield a measure quality product and that can be delivered to customer.

3.1 TDMBD in Action

This system is combining two approaches as shown in following flowchart. Initially requirement gathering and planning of process is done. After planning process as per test driven development test cases are written and executed. Now it depends upon the success of that test case whether further process move ahead, or code will be refactored. As soon as test succeeds DevOps process takes the lead. Fig. 3 shows the process flow of proposed framework where entire process of proposed approach is elaborated.

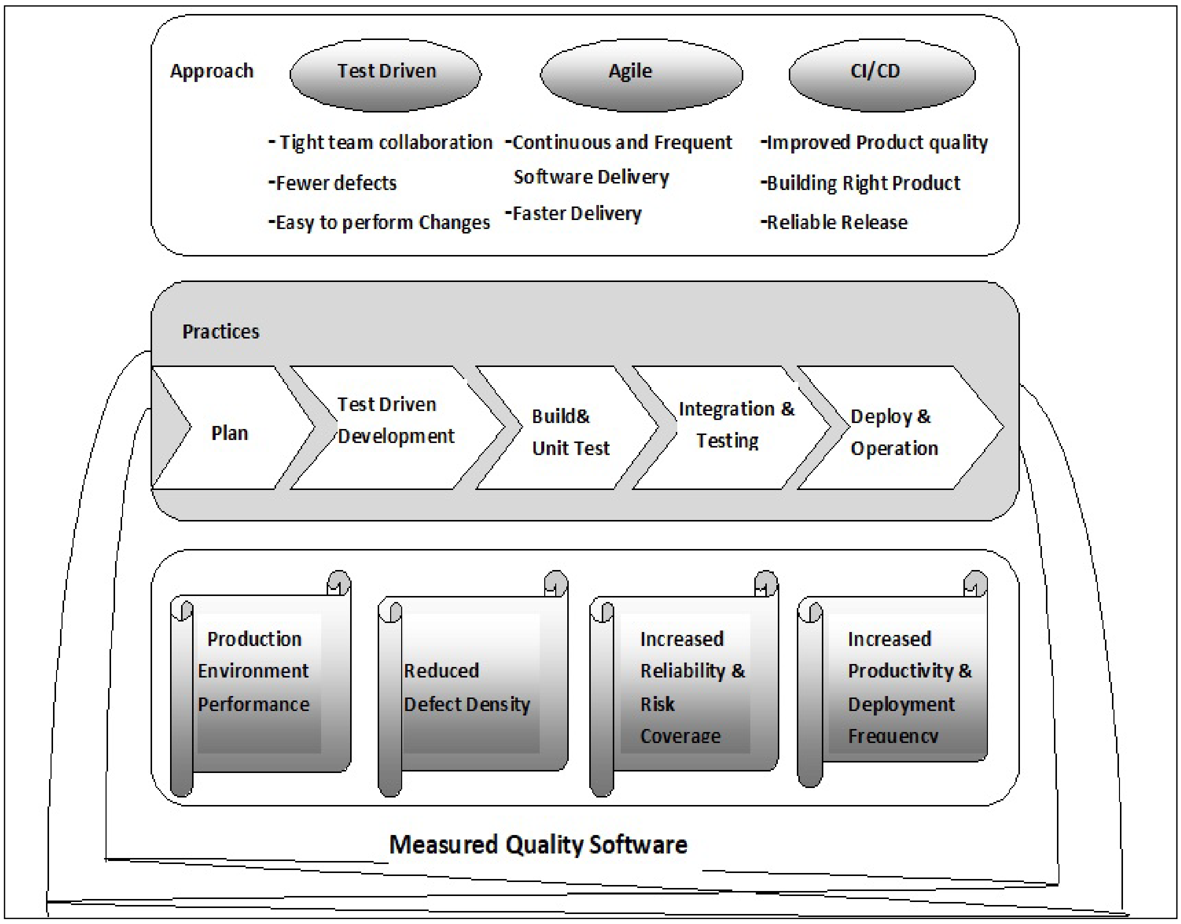

Fig. 2. Proposed framework (TDMBD)

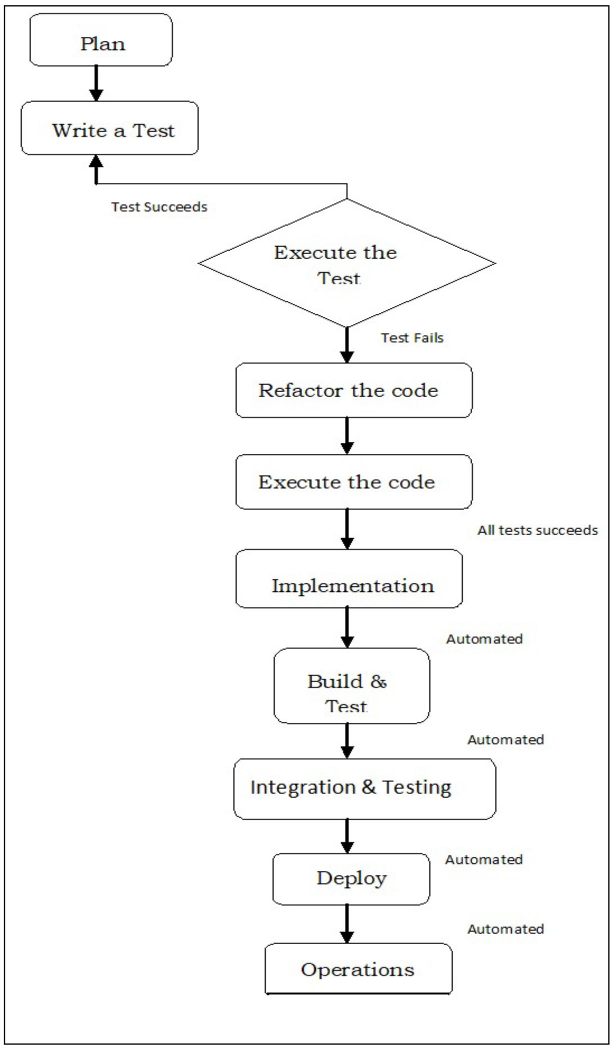

Fig. 3. Process flow of TDMBD approach

To implement TDMBD Jenkins tool is chosen. As Jenkins is an open-source tool and allows continuous integration and continuous delivery of projects. Some plug-ins are installed for smooth delivery of software. GitHub is used for code management; Build would be done by Maven. JUnit is used for test writing and execution. Deployment would be done by AWS and CHEF would take care of operational procedure. As shown in figure of process flow graph TDMBD approach start with test driven environment by writing and executing in JUnit plug-in. Further code is written in local repository and uploaded in GitHub which is already connected to Jenkins. In next step Maven will create build which contains a detailed description of our project, including information about dependencies, versioning and configuration management, application resources, team structure. Now again JUnit takes the charge to perform unit and integration testing. Further continuous integration, release tasks are handled by Jenkins itself and continuous deployment is done through AWS, chef plug-in take care of operation procedure. Chef plug-in easily setup, deploy and configure our project in AWS environment. Whole process is automated through Jenkins as no human intervention is required.

3.2 Selection of Metrics to Analyse the Quality of Software

Software systems quality is always judged by essential feature that is quality. Software quality feature not only proves excellence of just software system but also software process and components too. Level of accuracy and timely completion of task can be measure of quality but qualifiable viewpoint can be different so as software metric. Software metrics can be categorized into three ways, product based, project based and process based software quality metrics are a subset of software metrics that focus on the quality aspects of the product, process, and project (Kan, 2002). Product based metrics involves reliability, risk calculation, customer satisfaction, customer issues, performance measures etc. (Kumar and Yadav, 2013). Project level metrics relates to issues regarding projects like number of defects, time taken in overall development etc. (Futong and Tingting, 2013). Process metrics includes productivity, cycle time and many more (Dissanayake, 2018). We have chosen some of the software quality metrics from above mentioned literature for validation purpose. Desirable values of selected metrics are shown in Table1.

4. IMPLEMENTATION & METRICS EVALUATION

Entire frameworks process flow is implemented to elaborate scaling of TDD approach to DevOps development tasks. All test cases were written in JUnit inserted as plug-in to Jenkins tool.

In this work numerous component-based metrics are defined and evaluated. Most of the metrics discussed so far relies on estimating various quality attributes of source code. Such metrics are defined without considering how underlying concepts are determined and their relationships are identified to develop metrics. Therefore, considering these issues a suite of five metrics is defined in this work for assessing the performance of individual component as well as the whole system. Metrics mentioned in Table 1 are defined in following manner.

Component reliability measure (CRM)

A software component can be a quality component if it maintains the relationship between the methods and properties. Theproposed metric CRM is based on the reliability metric defined by Musa and Okumoto (1984). Software reliability can be defined as probability that software will not fail and will work as per requirement of customer in a specified environment and for a particular time. The probability of failure is calculated by testing a

Table 1. Metrics analysis

Type of metric	Subcategory of metric	Assigned value
Product based	Reliability	High
	Risk coverage	High
Project based	Defect density	Low
	Deployment frequency	High
Process based	Productivity (Efficiency)	High

sample of all available input states. Above mentioned model is used to derive our metric CRM. Jelinski Miranda reliability model describes the process reliability. But in case of proposed metric reliability of each component is evaluated and sum up to produce average reliability measure of whole process.

Expression for component reliability measure as below,

$CRM_x(t) = e^{-\sigma t_x}$ (1)

Where

$\sigma (t_x) = \phi [N-(x-1)]$ (2)

ϕ = a constant shows the failure rate of each fault per unit time

N = number of errors in the software

t_x = the time between (x-1)th and (x)th failure

Mean time failure function (σ) = $1/ \phi [N-(x-1)]$ (3)

And

$\sigma = 1/ \sigma_1 + 1/ \sigma_2 + 1/ \sigma_3 + \ldots\ldots + 1/ \sigma_N$ (4)

Mean time failure function will be reciprocal sum of failure rates of each system component. Using Equation(1), Equation(2), Equation(3) and Equation(4)

ORM (t) = $\mathbf{1 - \sum_{x=1}^{n} CRMx(t)}$

DevOps process components are running parallel. According to metrics analysis table, reliability should be high. So as CRM value should be high in proposed framework approach.

Component Risk Coverage Measure (CRCM)

Risk Coverage provides fast and accurate assessment of risk associated to latest release that is ready to go in production (Platz, 2020). It will be helpful in aligning test activities with customer's risk objectives. Risk coverage artifact would drive the percentage of business risk that is covered by test cases. According to test case prioritization, test that has high priority will get more weightage for associated risks.

Expression for component risk coverage measure as below,

CRCM= $\sum_{x=1}^{n} Wx$

Where n = no. of requirements

For overall risk coverage measure (ORCM) =, M > 0

Where M = total no. of components

Component Defect Density Measure (CDDM)

Defects are inevitable in any software. Some defects are not much considerable, but developer needs to keep an eye on every spike as it may ruin the overall performance of software. So defect density decides whether a software component is ready to deploy or not (Rahmani and Khazanchi, 2010). It also affects the overall quality of software. Defects may be of many types like issues in code after deployment, productions issues like database connectivity and many more.

CDDM = $\frac{\textit{Number of Defects}(D)}{\textit{Size of Release in KLOC}(S)}$

Overall Defect Density (ODDM) = $\sum_{i=1}^{n} \boldsymbol{CDDMn}$, n > 0

where n = no. of components.

Component Deployment frequency measure (CDFM)

Fast iteration and continuous delivery are the reasons behind the success of DevOps. Continuous deployment is next key measurement (Duvall, 2018). After how much time and for how long deployment process continues includes in mentioned process. Component deployment metric is associated with deployment stage of DevOps process. Smaller size deployments make it easy to test and deploy as well. Therefore, more releases with small updation is always better than less releases with high amount of updation. Further adding benefits of frequent deployment are early discovery of errors and repairing the same in earlier stages. Component deployment frequency is deployment of every component per unit time as shown in following equation.

CDFM = $\frac{\textit{Number of Deployment}(De)}{\textit{Unit of Time}(T)}$

CDFM = $\frac{\mathbf{De}}{\mathbf{T}}$

For Overall Deployment Frequency Measure (ODFM) = $\sum_{i=1}^{n} \boldsymbol{CDFMi}$, n > 0

where n = no. of components.

High deployment frequency is a good sign that overall functioning of software is smooth. Higher deployment frequency is directly proportional to higher efficiency. As per metric discussion component deployment frequency measure is equal to deployment per unit time. Time unit depends upon the size of project. If size of project is large or line of code (LOC) is high, then we can take unit time in number of weeks. But as size of data set is low, unit of time is considered in hours.

Component Productivity Measure (CPM)

Productivity is measured by throughput of process. For our process throughput is defined as units of work done within a set period of time (Batra and Jatain, 2020b). It is a measurement of developer's activity. By measuring throughput, we can track not only the details of delivery but also rate of success. That is why it yields into productivity.

Table 2. Data set with various measures

Sr. No.	Project name	Size (in LOC)	Components	Domain
1	Website of school	2031	15	Web application
2	Dome	1785	13	Graphics
3	She safe	1625	10	Application
4	Dhoondho	1680	10	Search application
5	Abridge	1983	13	Tool

Table 3. Component reliability analysis for existing approach

Project No.	No. of components	Input for x	Input for N	CRM
1	15	15	15	2.718
2	13	11	13	0.100
3	10	6	10	0.04
4	10	6	10	0.04
5	13	11	13	0.100

Table 4. Component reliability analysis for TDMBD approach

Project No.	No. of components	Input for x	Input for N	CRM
1	15	9	12	7.389
2	13	10	10	2.718
3	10	8	9	1.000
4	10	8	9	1.000
5	13	10	10	2.718

Efficiency (E) = $\frac{Number\ \ of\ \ User\ \ Stories\ \ of\ \ Component(Us)}{Time\ \ consume\ \ to\ \ fulfil(T)}$

Unit of work done can be measured in user stories. User stories completed by developer in given time will give us efficiency rate.

Overall productivity measure (OPM) = $\sum_{x=1}^{n} CPMx$ where n = no. of components

Overall quality of software (OQS) = ORM + ORCM + (-ODDM) + ODFM + OPM

As defect density should be low so we shall reduce it from overall quality calculation.

Data Set

As we had to demonstrate comparative analysis of DevOps to evaluate metrics so it was not possible to work on existing case studies. We have created five applications of different domains and different attributes to evaluate our framework. Table 2 shows the descriptive measures of various applications. Development environment for all of the applications is Java.

5. RESULTS AND DISCUSSION

Metric analysis of each component of software and overall software is done before and after migration. Experimental results of various artifacts are shown for all projects that were mentioned in datasets for different environment. Various projects have different number of components on which our artifacts are validated.

Feature analysis of existing and TDMBD approach using CRM. Table 3 shows the component reliability measure of existing approach where failure rate constant's value is fixed in each component.

Table 4 shows the component reliability analysis for proposed approach. Reliability is generally measure of accuracy. As data calculated from method mentioned in above section, reliability is high in our approach as compared to existing approach.

Results are clearly shown in Fig. 4 where our approach provides better reliability than the existing one. High reliability results into better performance that is system can perform correct for defined period as reliability follows law of exponential failure.

Fig. 4 elaborated more precisely the better reliability of our approach to existing approach in graphical presentation. Reliability of all projects mentioned in our data set is shown in following bar graph.

Feature Analysis of existing and TDMBD approach using CRCM

Table 5 shows the component risk measure for existing approach that consist total risk test amount. There are various tests that are broken, not tested, not even executed. As we can see that risk test those are positively executed is much in amount in Table 5.

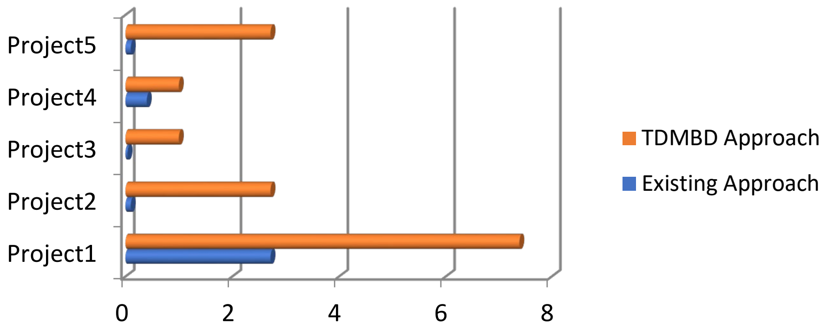

Fig. 4. Comparative analysis of CRM of existing and TDMBD approach

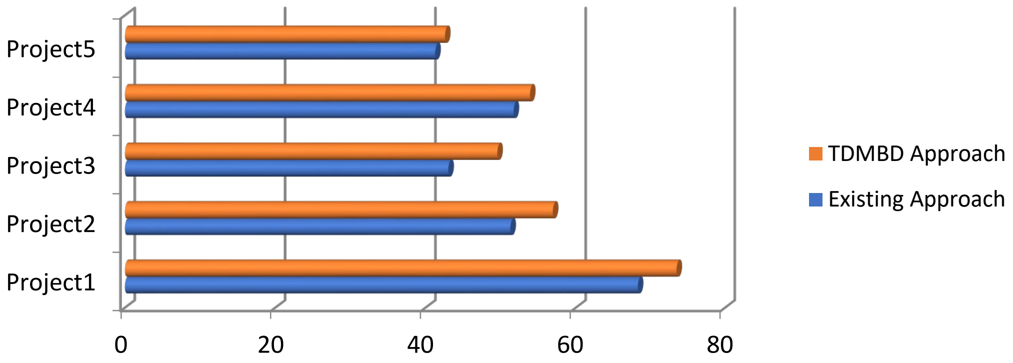

Fig. 5. Comparative analysis of CRCM of existing and TDMBD approach

Table 5. Component risk measure for existing approach

Project No.	Risks not tested	Risk broken	Risk tests not executed	Risk tests executed positively	Total risk test	Percentage of risk coverage
1	18	35	20	157	230	68.26%
2	15	28	14	118	175	51.30%
3	13	26	12	99	150	43.04%
4	8	17	16	119	160	51.73%
5	9	12	14	95	130	41.30%

Table 6. Component risk measure for TDMBD approach

Project No.	Risks not tested	Risk broken	Risk tests not executed	Risk tests executed positively	Total risk test	Percentage of risk coverage
1	20	19	20	169	230	73.47%
2	14	16	14	131	175	56.95%
3	7	17	12	114	150	49.56%
4	7	13	16	124	160	53.91%
5	8	10	14	98	130	42.60%

Table 6 shows the details of risk coverage analysis of proposed approach. Amount of positively executed risks test are higher in comparison to existing approach. But defined artefact relates to percentage coverage which will provide better instinct to risk.

Fig. 5 shows the percentage coverage of both approaches for overall risk coverage measure. TBMBD approach ensures better risk coverage tends to increase customer satisfaction. Risk coverage not only ensures faster delivery but also provides timely delivery of product.

Table 7. Component defect density analysis of existing framework

Project	Components	Size (LOC)	No. of defects	Defect density (defect/KLOC)
1	15	2031	30	14.77
2	13	1785	17	9.52
3	10	1625	13	8.00
4	10	1680	14	8.33
5	13	1983	20	10.08

KLOC = Kilo Line of Code

Table 8. Component defect density analysis of TDMBD approach

Project	Components	Size (LOC)	No. of defects	Defect density (defect/KLOC)
1	15	2031	15	7.38
2	13	1785	10	5.60
3	10	1625	7	4.30
4	10	1680	8	4.76
5	13	1983	10	5.04

Table 9. Deployment frequency of existing process

Project No.	Size of project (LOC count)	Number of deployments	Time taken in hr.	Deployment frequency per hour
1	2031	15	1.5	10.00
2	1785	12	1.2	10.00
3	1625	11	1	11.00
4	1680	11	1	11.00
5	1983	12	1.3	9.23

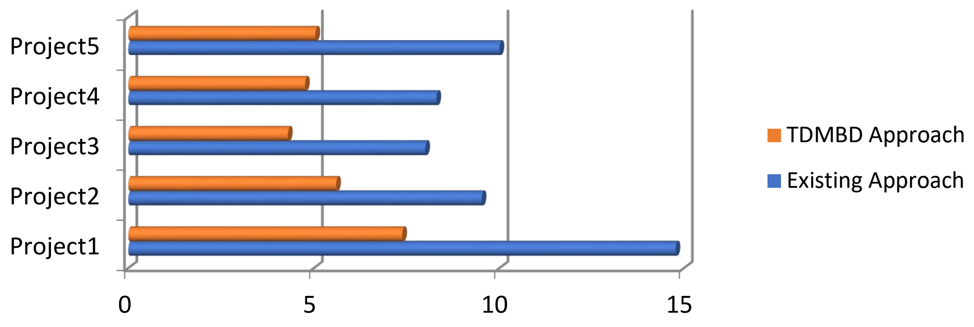

Fig. 6. Comparative analysis of CDDM of existing and TDMBD approach

Feature Analysis of existing and TDMBD approach using CDDM

Table 7 presents component defect density analysis of existing approach. Defect density metric gives the results by dividing the defects by size of component as defined in above section.

Table 8 shows component defect density analysis of new approach. After calculating defect density comparison clearly shows that in TDMBD approach defect density is lower than existing one.

Fig. 6 shows the comparative analysis of defect densities of both approaches. CDDM of TDMBD approach is less than existing approach which yields good quality of software.

Feature analysis of existing and TDMBD approach using CDFM

Table 9 consists of data related to deployment frequency of each component of existing framework. Clearly it is shown that much time is taken every deployment that results into less deployment frequency.

While Table 10 shows deployment frequency data details of TDMBD approach. Here time taken to deploy components is less and no. of deployments are even higher resulting high deployment frequencies.

Table 10. Deployment frequency of TDMBD approach

Project No.	Size of project (LOC count)	Number of deployments	Time taken in hr.	Deployment frequency per hour
1	2031	17	1.1	15.45
2	1785	14	0.9	15.55
3	1625	14	0.7	20.00
4	1680	13	0.8	16.25
5	1983	15	0.9	16.66

Table 11. Component productivity measure of existing process

Project No.	LOC	Number of user story (total)	Time (in weeks)	Throughput
1	2031	150	20	7.5
2	1785	135	17	7.94
3	1625	120	15	8
4	1680	125	16	7.81
5	1983	140	18	7.77

Table 12. Deployment frequency of TDMBD approach

Project No.	LOC	Number of user story (total)	Time (in weeks)	Throughput
1	2031	150	15	15
2	1785	135	13	13.5
3	1625	120	12	12
4	1680	125	12	12.5
5	1983	140	14	14

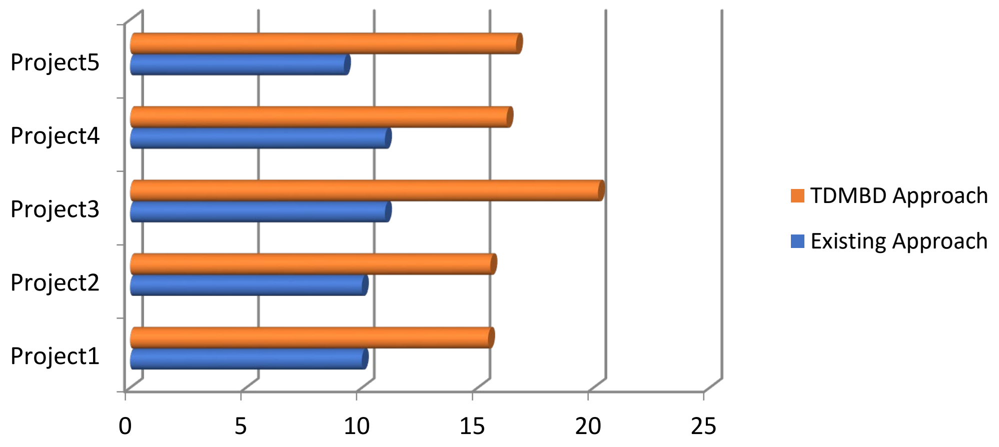

Fig. 7. Comparative analysis of CDFM of existing and TDMBD approach

Fig. 7 shows the graphical presentation of validation of defined artifact to the existing approach. Results clearly show that CDFM of proposed approach is better than the existing one. In TDMBD approach deployments are more frequent after regular intervals in comparison to traditional approach where frequency of deploying components after long interval of time.

Feature Analysis of existing and TDMBD approach using CPM

Table 11 shows the component productivity measure of existing framework. As discussed in above section productivity is measured by throughput and throughput is derived by user stories divided by time consumed to accomplish those user stories.

As clearly shown in Table 12 which contains data analysis of productivity measure of TDMBD approach that throughput is increased in our approach whereas decreased up to the level of almost half in existing approach.

Fig. 8 shows the clear comparative analysis of CDFM of both approaches. It can be concluded that new approach presents better productivity in terms of efficiency.

Table 13 presents the conclusive results for all defined

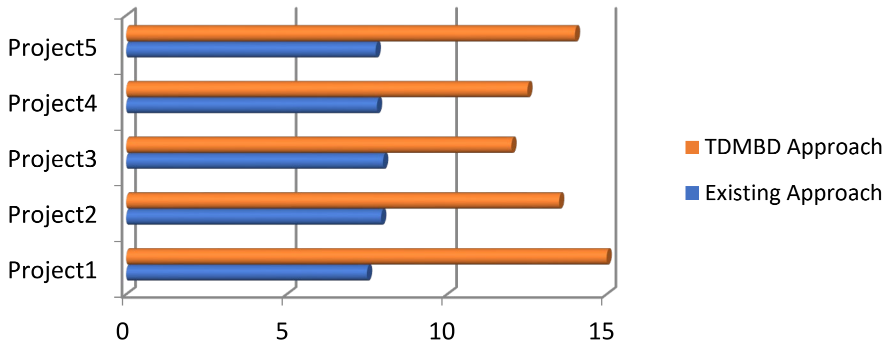

Fig. 8. Comparative analysis of CDFM of existing and TDMBD approach

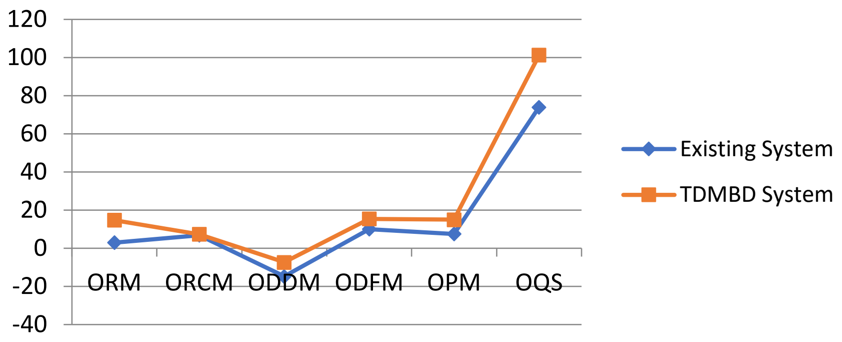

Fig. 9. Overall quality metrics comparison of both approaches

Table 13. Metric level analysis of both approaches

Metrics	Existing system	TDMBD system
ORM	2.98	14.7
ORCM	68.26	73.47
ODDM	14.77	7.38
ODFM	10	15.45
OPM	7.5	15
OQS	73.97	111.24

metrics of new and traditional approach. It provides the comparative analysis between the metrics generated for each component of individual product. As described in dataset 5 projects have different number of components and artifacts are calculated for different components. Finally overall quality measures of all components are calculated.

TDMBD system gives better results as shown in above table for every metrics. Overall measure of quality also provides better results. Graphical visualization is done in Fig. 9 to demonstrate enhanced quality of our approach to existing one.

As Fig. 9 depicts overall quality of software when TDMBD approach is applied is better than the existing approach. Defect density metric should be low as can be mapped through Table 1 for its desirable value that is why it kept negative in visual presentation, and it will be reduced in overall quality calculation.

6. CONCLUSION & FUTURE WORK

This research focuses on test driven DevOps framework which is used to transit the existing system to new software. The presented research work is the combination of two popular approaches of software development. This composite framework consists of test-driven development approach along with DevOps approach. The proposed TDMBD model has taken the test-driven development system as input and applied a series of DevOps process. To confirm the validity of the methodology, a suite of metrics is introduced as artifacts and those artifacts are identified at process, project, and product level. To achieve this stage, software components are evaluated, and qualitative observations are received. As committed by proposed approach quality standards got increased and performance issue are resolved as all metrics defined produced better results as compared to existing approach. In future we can work on real time large industry level project where our own artifacts can be challenged and also an automated tool can be designed to ease the process of transition. We can also work on monitoring and security features of software.

REFERENCES

Astel, D. 2003. Test driven development: A practical guide (A. D. Library (ed.)). Prentice Hall Professional Technical Reference.

Batra, P., Jatain, A. 2020a. DevOps: Current practices, challenges and implications. International Journal of Advanced Sciences and Technology, 29, 11991–12001. http://sersc.org/journals/index.php/IJAST/article/view/27879

Batra, P., Jatain, A. 2020b. Measurement based performmace evaluation of DevOps. Lecture Series on Computational Performance Evaluation.

Beck, K., Beedle, M., Bennekum, A. Van, Cockburn, A. 2001. Manifesto for agile software development. https://pdfs.semanticscholar.org/2dc5/d5a781ab55d3bba09d2fdb05ebf87bde7a2f.pdf

Dissanayake, S. 2018. Measurable metrics for software process improvement. European Journal of Computer Science and Information Technology, 6, 33–43.

Duvall, P. 2018. Measuring DevOps success with four key metrics. https://stelligent.com/2018/12/21/measuring-devops-success-with-four-key-metrics/

Elberzhager, F., Arif, T., Naab, M., Süß, I., Koban, S. 2017. From agile development to devops: Going towards faster releases at high quality - Experiences from an industrial context. Lecture Notes in Business Information Processing, 269, 33–44. https://doi.org/10.1007/978-3-319-49421-0_3

Erich, F., Amrit, C., Daneva, M. 2014. A mapping study on cooperation between information system development and operations. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8892, 277–280. https://doi.org/10.1007/978-3-319-13835-0_21

Floris, E., Amrit, C., Daneva, M. 2014. DevOps litterature review. Product-Focused Software Process Improvement, 8892. https://doi.org/10.1007/978-3-319-13835-0

Futong, H., Tingting, S. 2013. Software project metrics and quality management. Proceedings - 2013 9th International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP 2013. https://doi.org/10.1109/IIH-MSP.2013.158

Gottesheim, W. 2015. Challenges, benefits and best practices of performance focused DevOps. LT 2015 - Proceedings of the 4th ACM/SPEC International Workshop on Large-Scale Testing, in Conjunction with ICPE 2015, 3. https://doi.org/10.1145/2693182.2693187

Huttermann, M. 2012. Beginning DevOps fpr devlopers. In DevOps for Developers (2012th ed., 4–13). Apress.

Hüttermann, M. 2012. Introducing DevOps. In DevOps for Developers, 15–32. https://doi.org/10.1007/978-1-4302-4570-4_2

Khan, S.H. 2002. Software quality metrics overview. Metrics and Models in Software Quality Engineering. Boston,2nd edition, Addison-Wesley.

Kumar, C., Yadav, D.K. 2013. Software quality modeling using metrics of early artifacts. IET Conference Publications. https://doi.org/10.1049/cp.2013.2285

Liu, Y., Zhou, Y. 2017. The challenges and mitigation strategies of using DevOps during software development. Blekinge Institute of Technology.

Madeyski, L., Szała, Ł. 2007. The impact of test-driven development on software development productivity - An empirical study. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/978-3-540-75381-0_18

Mäkinen, S., Münch, J. 2014. Effects of test-driven development: A comparative analysis of empirical studies. Lecture Notes in Business Information Processing. https://doi.org/10.1007/978-3-319-03602-1_10

Musa, J.D., Okumoto, K. 1984. A logarithmic poisson execution time model for software reliability measurement. In Citeseer. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.111.2201&rep=rep1&type=pdf

Nagarajan, A.D., Overbeek, S.J. 2018. A DevOps implementation framework for large agile-based financial organizations. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11229 LNCS, 172–188. https://doi.org/10.1007/978-3-030-02610-3_10

Nicolau de França, B.B., Jeronimo, H., Travassos, G.H. 2016. Characterizing DevOps by hearing multiple voices. ACM International Conference Proceeding Series, 53–62. https://doi.org/10.1145/2973839.2973845

Platz, W. 2020. Risk coverage: A new currency for testing. https://www.stickyminds.com/article/risk-coverage-new-currency-testing

Rahmani, C., Khazanchi, D. 2010. A study on defect density of open source software. Proceedings - 9th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2010. https://doi.org/10.1109/ICIS.2010.11

Shahin, M., Ali Babar, M., Zhu, L. 2017. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. IEEE Access, 5, 3909–3943. https://doi.org/10.1109/ACCESS.2017.2685629