Metric based Performance Measurement of Software Development Methodologies from Traditional to DevOps Automation Culture

Poonam Narang[†] and Pooja Mittal[††],

Maharshi Dayanand University, Rohtak, Haryana, India

Summary
Successful implementations of DevOps practices significantly improvise software efficiency, collaboration and security. Most of the organizations are adopting DevOps for faster and quality software delivery. DevOps brings development and operation teams together to overcome all kind of communication gaps responsible for software failures. It relies on different sets of alternative tools to automate the tasks of continuous integration, testing, delivery, deployment and monitoring. Although DevOps is followed for being very reliable and responsible environment for quality software delivery yet it lacks many quantifiable aspects to prove it on the top of other traditional and agile development methods. This research evaluates quantitative performance of DevOps and traditional/ agile development methods based on software metrics. This research includes three sample projects or code repositories to quantify the results and for DevOps integrated selective tool chain; current research considers our earlier proposed and implemented DevOps hybrid model of integrated automation tools. For result discussion and validation, tabular and graphical comparisons have also been included to retrieve best performer model. This comparative and evaluative research will be of much advantage to our young researchers/ students to get well versed with automotive environment of DevOps, latest emerging buzzword of development industries.

Keywords:
Automation, Automation Tools, DevOps, Software Development, Tool chains

1. Introduction

Software development has covered many methodologies from traditional and agile to DevOps. Traditional development methods like waterfall, iterative, spiral, prototype etc has agonized with many flaws responsible for late and over budget delivery of software. Fast and successful delivery was one of the major challenges for traditional methods. Agile methods came up with better solutions of speedy releases in terms of small sprint sizes. Agile approaches introduced. agility in their methods but still lack continuity in development and operations where DevOps comes in picture. DevOps relies on 5Cs of software development. These are Continuous Integration, Continuous Testing, Continuous Delivery, Continuous Deployment and Continuous Monitoring. The continuous environment in DevOps culture is achieved through different set of alternative tools. With the passage of time, development industries realized that recent or latest software delivery approaches or methods to be much better than traditional ones. DevOps, being the new, emerging and latest software development culture, outperforms other traditional and agile methods in every aspect. Following figure represents working principles of DevOps –

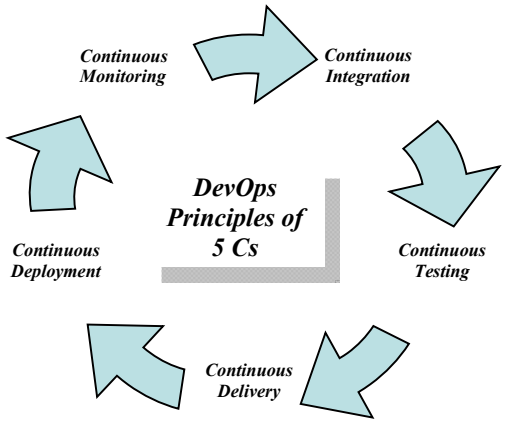

Figure1. DevOps principles of continuous software development stages

As depicted in the above figure (1), DevOps works on the principles of infinite continuous cycle that starts from continuous integration of code repositories and continues till the monitoring stage of the software development. DevOps undoubtedly is the latest buzzword of development industries but still it lacks in quantitative quality evaluation. This research quantitatively evaluates the performance of different software development methodologies including DevOps based on software metrics measurement. For the application of these metrics and to get quantitative results, three sample code repositories have been taken into consideration. This paper also considers our already proposed [1] and implemented [2] DevOps hybrid automation tool chain model. The result of this performance evaluation validates the quality of delivered product and also helps to retrieve the best performer development methodology. The outcome of current research work will be of useful to our young researchers/ students to understand course of action of DevOps and automation tools. It will also be of great help to software developers to select best performer tools from

Manuscript received June 5, 2023
Manuscript revised June 20, 2023
https://doi.org/10.22937/IJCSNS.2023.23.6.13

alternative sets of automation tools available.
The remaining sections of the research paper are organized as – first section introduces the DevOps working principles followed by literature study section. Next section defines the metrics that are evaluated to find the performance of development methodologies along with their tabular and graphical comparisons. Last section concludes the work with future or further work.

2. Related Study

SDLC or Software Development Life Cycle involves different phases or stages in software development. SDLC contains detailed elaborations for different steps or procedures necessary to formulate software. It covers many methodologies like traditional, agile and currently, DevOps These methodologies cover various models under them. For example, traditional methodologies include Incremental, evolutionary, Waterfall, V Model, spiral etc whereas agile covers Scrum, Kanban, XP, RAD etc. DevOps, on the hand, involves set of alternative automation tools to reach goals of being competitive in the era of digital transformation and increasing their velocity to adapt or react to change.[3] DevOps, with its continuity principles, ease the path of organizations to withstand in this competitive world. Main focus of current research is the description and comparison of DevOps with other existing methodologies.

2.1 Traditional Methodologies

Traditional methodologies include specifically waterfall model introduced first by Winston W. Royce in 1970s. His paper [4] clearly describes stage wise waterfall model. Through these stages including iterative approaches, the author introduced the development of large systems. Step by step approach of traditional methodologies allows large size projects to handle and deliver successfully. But on the other side, traditional development methods are inflexible and also fail to respond on aggressive or frequently changing requests of customer. [5] In comparison to traditional, agile development methodology provides set of practices that allow quick adaptations of changing customer needs

2.2 Agile Development Methods

Although agile methods come up with coping up all limitations or flaws of traditional development methodologies and these are well proven for small, collocated teams but authors in their research [5] confirms that agile methods also work for large sized and distributed projects. Another similar research on agile methods [6] also proved agile methods for inspiring in large and very large-scale development. So, agile methods favor more communication, continuous integration along with rapid product delivery through iterative and incremental approach, but at the same time agile methods suffer from many limitations of planning lacks, documentation lacks along with lack of predictability etc. [7] Authors conducted online survey to find actual limitations of agile methods beyond the existing literature survey. DevOps, here, comes up with one of the proposed solution to many development and delivery pressure including quality of developed product. [8]

2.3 DevOps and its Existing Models

Alok Mishra and Ziadoon Otaiwi [8] in their work on DevOps, analyzes implications of DevOps features on software quality. Primary focus of DevOps is to increase deployment speed, frequency and product quality. DevOps, Development and Operations, bridges all kinds of communication gaps between dev and ops teams with the targets of reducing discrepancies of these teams. [9] In another research on case study of five companies, authors [10] agree upon well coordination between development and operations teams to deliver or deploy quality products. Although multiples of research agrees to adopt DevOps, yet at the same time, many theories are against DevOps and talk about lack of quantification of quality and performance measure [11]. Another research on case study on DevOps agreed DevOps help companies in reaching their goals and also increased their velocity in react to change [13].

Literature also confirms the existence and successful implementation of different DevOps Models in practice. [12] Similarly, many other papers including [13] [14], accepts the emerging paradigm as a response to growing knowledge of existing many types of communicational or collaboration, cultural gaps between dev and ops teams functions.

2.4 Motivation

Our motivation behind this research is to address the challenge of quantification of DevOps quality and timeliness of delivered product. For this purpose, current research also considers our already proposed [1] and implemented [2] hybrid automation tools model. On the basis of existing literature and our tools model, this work performs evaluation of different types of parameters for quality measurement. This research has taken three sample software applications based or developed in JDK

environment for measuring DevOps performance and quality.

3. Hybrid Model for DevOps

For the quality validation of DevOps tools, we have considered our previously proposed [1] and implemented hybrid model [2] for DevOps for integrated tool chain (ITC) as depicted in the following figure –

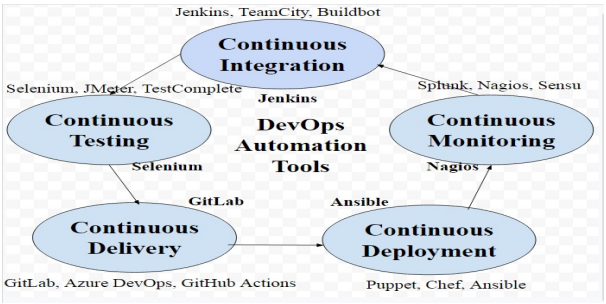

Figure 2. Hybrid model for DevOps automation toolset [1]

As shown in above figure (2) for hybrid model, different tools are proposed or selected as best performer tool based on performance evaluators. Current research work considers same set of automation tools for DevOps implementation.

4. Research Design

For this underlying research, we have considered three sample java based applications - Web Site for Online Faculty Recruitment, Car searching, scientific calculator tool. These applications or code is designed in local repositories in JDK environment and uploaded in GitHub to make it remote repository and for the smooth working of DevOps environment. After writing and uploading of code, next is to plan and write test cases for proper execution and implementation of the code. If the test case fails then we are to rewrite the test case after code refactoring. Successful test case execution is then followed by code deploy and monitoring or operations phase of the project. The complete step by step procedure in terms of process flow diagram for current research work is shown below –

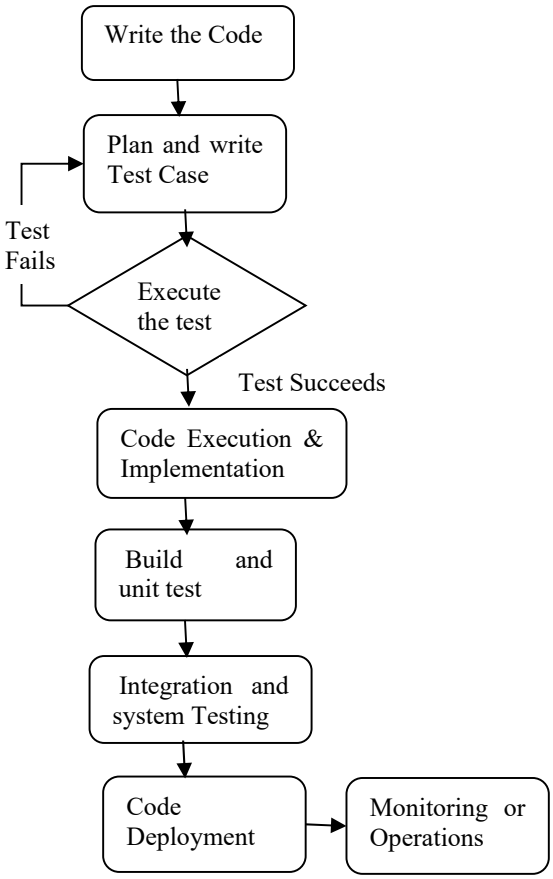

Figure 3. Process flow diagram of current research work

As clearly depicted in above flowchart or diagram (3), writing test cases and successful code testing is the major part of the DevOps process. For the implementation purpose, Jenkins continuous integration tool is selected which also takes charge of continuous delivery of the product. Different Plugins are also installed with Jenkins for the smooth working the tool. Code repository or maintenance is done through GitHub. Ansible with Jenkins is responsible for continuous deployment of the code. Test cases are written using Junit. Build work is done by Maven. All these steps or procedure is fully automated and done through DevOps automation tool set.

5. Quality Evaluation of DevOps and other Development Methodologies through Metric Measurement

Quality plays important role in software acceptance. Software quality is not a single factor or value but it covers many different parameters like testability, predictability, maintainability etc to consider for achieving the complete quality product. So quantification of these performance or efficiency parameters becomes more important for

measuring the quality value. This research work considers all three types of software metrics – Project, Process and Product metrics. Project metrics covered in current work are Project Defect Density and Release Deployment Frequency, to measure defect density covered in project along with the deployment frequency of the project in terms of releases. Similarly, Process metrics covered productivity of the whole process in the form of throughput of the system. Lastly, product, metrics involves identification of risk involved and reliability of the developed product. These software metrics along with their expected outcome is shown in following table –

TABLE1. SOFTWARE METRIC CLASSIFICATIONS FOR VALIDATION OF THE SOFTWARE DEVELOPED

Type of software metric	Software Metric	Expected Outcome or Results
Project	Project Defect Density	Low
	Release Deployment Frequency	High
Process	Risk Identification	High
Product	Process Productivity	High

Above categorization of Software metrics in table (1), clearly mentions expected results of the metric. These metrics with their calculation formulas and methods are explained below –

5.1 Project Defect Density (PDD)

Project defect density refers to the deploy readiness of the software that is whether software can be deployed or not. PDD in actual depends directly on presence of defects in the system. As defects can incur at any stage of software development, so checks at regular intervals become necessary activity of development. The value of PDD must be low to ensure quality delivery of the software. Defect density formula is given as –

$$\text{PDD} = \sum_{i=1}^{n} \frac{TotalNumberofDefects}{SizeofSoftware(inKLOC)} \qquad (1)$$

Above formula (1) is used to first find defect density of individual components or modules of the system and to get the defect density for the whole system/project by summing them up. Here, n refers to total number of components or modules in the system and n>0.

5.2 Release Deployment Frequency (RDF)

Release deployment frequency tells total number of deployments in a particular time period. In other words, RDF refers to the rate of release deployment. Higher the value of RDF, lesser is the chance of errors/ defects in the system. Formula for calculating deployment frequency is given as –

$$\text{RDF} = \sum_{i=1}^{n} \frac{TotalNumberofDeployments}{TimeUnit(inHours)} \qquad (2)$$

Formula (2) above, calculates total number of deployments or release count in particular time unit, taken in hours, for individual components and adding them all to get the RDF for the whole system/project.

5.3 System Risk Identification (SRI)

System risk identification refers to the assessment of risk associated with the project/ system to be developed. High value of risk factor reflects identification of more risk components and ensures safe and risk free delivery of software. Expression or formula for system risk identification is given as –

$$\text{SRI} = \sum_{i-1}^{n} Wx \ , \ n>0 \qquad (3)$$

In above expression (3), W_x refers to the weightage assigned to each individual risk and summing up all to get system risk identification number. Here, n refers to total number of components in the system.

5.4 Process Productivity (PP)

Productivity of any system is total units of work done in particular time period. It refers to the throughput of the system. Process or system productivity is measures as –

$$\text{PP} = \sum_{i=1}^{n} \frac{TotalNumberofUserStories}{TimeTakenToComplete} \qquad (4)$$

Above formula (4) to find the productivity of the process or system, computes summation of productivity of individual components. User stories, here, refers to the unit of work done in given time period.

6. Data Set

Current research compares traditional and agile methodologies with DevOps development. For this purpose, three sample applications /projects in java have been designed through traditional methods and later on uploaded these local repositories to GitHub for

implementation of DevOps tools. Different parameters for measurement of applications are calculated as –

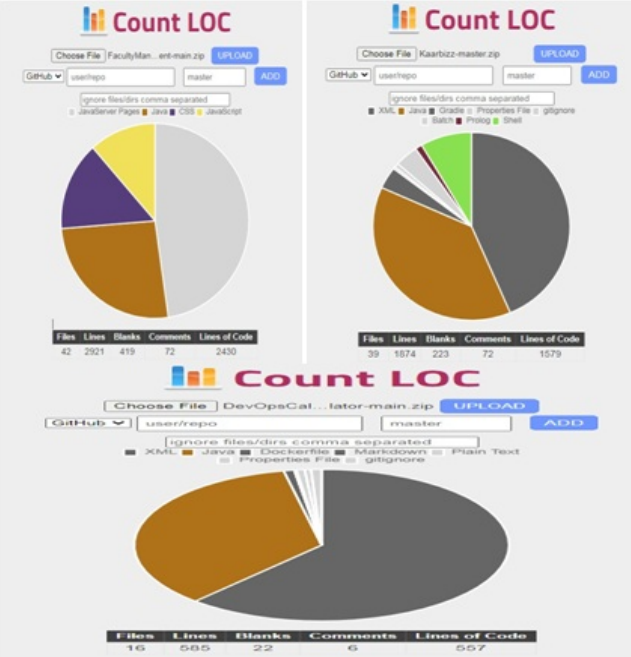

Figure 4. Components/Modules and LOC count for sample applications

Above Figure (4), gives total count of lines of code along with different components and other parameters to consider for metric evaluation purpose. Following table shows descriptive measures of these sample applications –

TABLE2. SAMPLE SOFTWARE APPLICATIONS AS DATA SET FOR CURRENT RESEARCH WORK

Src No	Name of Project/ Application	Size (LOC)	No of Components/ Modules	Application Domain
1	Web Site for Online Faculty recruitment (Project1)	2430	18	Web based
2	Car search Application (Project2)	1579	14	Search application
3	Scientific calculator tool (Project3)	557	8	Tool

Different sample applications mentioned in table (2) have Java as application development environment.

7. Results and Discussion

Software metric evaluation is an essential requirement for the measurement of project progress, successful delivery, deployment and operations of the whole process, project and product. Different metrics defined above are evaluated for the data set of table (2) and results are also discussed below.

7.1 Project Defect Density (PDD)

Defect Density is an important project metric to measure different defects included in the system in the form of bugs. Table 3 below shows the defect density measure of the whole project or system as calculated with traditional methods of development –

TABLE3. PDD MEASURE OF SAMPLE APPLICATIONS AS PER TRADITIONAL DEVELOPMENT APPROACHES

Sr No	Project	Size (LOC)	No of Components/ Modules	Total no of Defects	Defect Density (PDD)
1	Project1	2430	18	30	12.35
2	Project2	1579	14	18	11.40
3	Project3	557	8	10	17.95

Defect density is computed in above table (3) using traditional development approaches and by dividing total number of defects with size of the corresponding project as given be expression (1). Table 4 below computes defect density with DevOps development –

TABLE 4. PDD MEASURE OF SAMPLE APPLICATIONS AS PER DEVOPS DEVELOPMENT CULTURE

Sr No	Project	Size (LOC)	No of Components/ Modules	Total no of Defects	Defect Density (PDD)
1	Project1	2430	18	15	6.17
2	Project2	1579	14	8	5.07
3	Project3	557	8	4	7.18

As seen in table (4) above, DevOps culture of development has low value for defect density that clearly indicates more reliability and success of the underlying process or project. Following figure displays graphical comparison of these methods –

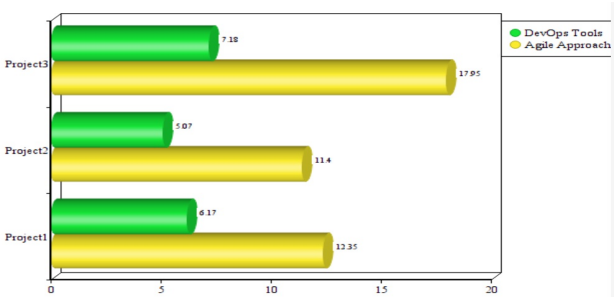

Figure 5. Graphical comparison of traditional and DevOps development culture for defect density measure

Above figure (5) indicates low value of defect density for DevOps development culture as compared to

traditional methods. DevOps ensures reliable, defect free and successful delivery of software.

7.2 Release Deployment Frequency (RDF)

Deployment frequency data of applications or java projects with the usage of traditional principles and rules is shown in table below –

TABLE 5. RDF MEASURE OF SAMPLE PROJECTS USING TRADITIONAL METHODS OF DEVELOPMENT

Sr No	Project	Size (LOC)	No of deployments	Time taken to deploy (hr)	Deployment Frequency (RDF)
1	Project1	2430	18	1.8	10
2	Project2	1579	15	1.5	10
3	Project3	557	9	1	9

Table (5) above contains the data related to deployment frequency in traditional methods and table below shows for DevOps development –

TABLE 6. RDF MEASURE OF SAMPLE PROJECTS USING DEVOPS DEVELOPMENT

Sr No	Project	Size (LOC)	No of deployments	Time taken to deploy (hr)	Deployment Frequency (RDF)
1	Project1	2430	20	1.1	18.18
2	Project2	1579	16	0.9	17.78
3	Project3	557	10	0.7	14.29

Above table (6) clearly indicates the high value of deployments that leads to acceptance to frequent changes to the system. Following figure also shows comparative graph of these two development methodologies –

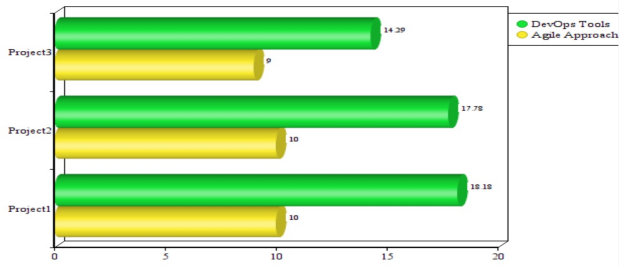

Figure 6. Comparative study graph for DevOps and Traditional approaches

As depicted in above graphical comparison figure (6), DevOps has higher number of deployments as compared to earlier approaches.

7.3 Risk Identification (RI)

Risk coverage analysis or risk identification gives the total number of risk sets injected with risk sets executed positively and total risks not tested or broken. Following table shows risk identification measurement as in the earlier development approaches –

TABLE 7. RI ESTIMATE OF SAMPLE APPLICATIONS USING TRADITIONAL METHODS OF DEVELOPMENT

Sr No	Project	Total Risk Tests	Total no of risk tests executed	Risks broken	Total Risks not tested	Risks not executed	Risk Coverage (%age)
1	Project1	275	188	42	22	23	68.36
2	Project2	150	104	23	12	11	37.81
3	Project3	60	40	9	5	6	14.55

Table (7) above depicts different risks set coverage as per earlier development approaches and table below contains data concerning with risk sets included and covered –

TABLE 8. RI ESTIMATE OF SAMPLE APPLICATIONS USING DEVOPS CULTURE

Sr No	Project	Total Risk Tests	Total no of risk tests executed	Risks broken	Total Risks not tested	Risks not executed	Risk Coverage (%age)
1	Project1	275	211	23	24	17	76.73
2	Project2	150	116	12	13	9	42.18
3	Project3	60	46	5	5	4	16.62

Above table (8), estimates or measures total number of risks covered by the underlying development approach. Graph beneath shows comparative study of risk sets coverage by both development approaches –

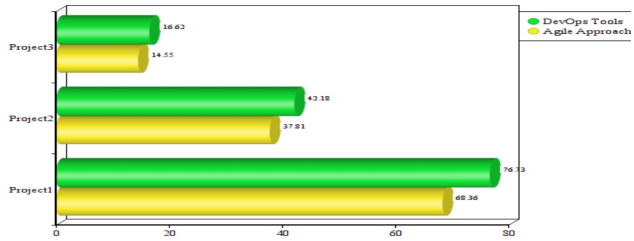

Figure 7. Risk coverage comparative study graph for DevOps and Traditional approaches

Above graph in figure (7) again clearly indicates more risk coverage by DevOps as compared with earlier development approaches.

7.4 Process Productivity (PP)

Process productivity also measured as throughput of whole system or software. It measures total units of work done in particular time period. PP for traditional methods are shown in the following table –

TABLE 9. PP ESTIMATE/ MEASURE OF CURRENT DATA SET USING TRADITIONAL METHODS

Sr No	Project	Size (LOC)	Total No of user stories	Time taken (in weeks)	Process Productivity (PP)
1	Project1	2430	180	18	10
2	Project2	1579	117	12	9.75
3	Project3	557	41	4	10.25

Table (9) above gives productivity measure in terms of throughput expression (4), it shows data for the traditional approaches. Similarly, following table includes the data for DevOps development culture –

Table10. PP Estimate/ Measure of current data set using DevOps methods

Sr No	Project	Size (LOC)	Total No of user stories	Time taken (in weeks)	Process Productivity (PP)
1	Project1	2430	180	14	12.86
2	Project2	1579	117	9	13
3	Project3	557	41	3	13.67

Above table (10) clearly indicates more throughput or productivity measure for DevOps culture. Following comparative graph gives more evidence for these tables –

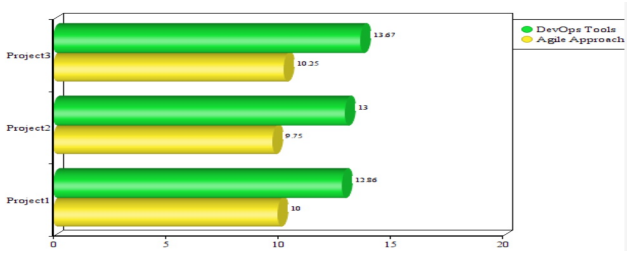

Figure 8. Graphical Comparison for Productivity measure of Traditional and DevOps Development approaches

In the above figure (8), DevOps shows speedy releases or output in terms of productivity measure in comparison with traditional approaches.

DevOps development culture gives better results as depicted in above tabular and graphical comparisons. Each of the performance metric and graphic visualization shows DevOps as symbol of quality and speedy delivery with less defect density and high coverage of risk sets.

8. Conclusions and Future Work

Main focus of current research work is the quality validation of DevOps development culture over traditional approaches including agile methods. To confirm the validity of DevOps quality, three java applications from different domain are considered along with DevOps performance metrics. Software components with different parameters are observed for these applications and values were calculated for considered metrics. Results, shown using tabular and graphical methods, confirm the quality validation of DevOps over other existing development approaches. As a part of future work, real time industry data can also be taken or our own tool can be designed to get more automated results.

REFERENCES

[1] Poonam Narang, Pooja Mittal, *Hybrid Model for Software Development: an Integrated Comparison of DevOps Automation Tools,* Indonesian Journal of Electrical Engineering and Computer Science (IJEECE), IAES Publishers, ISSN 2502-4752, Scopus indexed, SJR 2020 (Q3 0.241, (accepted for publication).

[2] Pooja Mittal, Poonam Narang, *Implementation of DevOps Hybrid Model for Project Management and Deployment using Jenkins with Plugins*, Journal of Information and Communication Technology (JICT), Scopus, ESCI, (Paper communicated).

[3] Marta Gomes, Ruben Pereira, Miguel Silva, Jose Braga de Vasconcelos, Alvaro Rocha, *KPIs for Evaluation of DevOps Teams,* World CIST2022, Information systems and Technology, pp 142-156, LNNS, Vol 470.

[4] Dr. Winston W. Royce at https://en.wikipedia.org/wiki/, Retrieved May 20, 2022,

[5] Georgios Papadopoulos, *Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects*, Elsevier Procedia – Social and Behavioral Sciences, Vol 175, Feb 2015, pp 455-463

[6] Torgeir Dingsoyr, Nils Brede Moe, Tor Erlend Faegri and Eva Amdahl Seim, *Exploring Software Development at the very large-scale: a revelatory case study and research agenda for agile method adaptation,* SpringerLink Empirical Software Engineering 23, 490- 520 (2018).

[7] Ashish Agrawal, Mohd, Auranzeb Atiq, L.S. Maurya, *A Current Study on the Limitation of Agile Methods in Industry Using Secure Google Forms*, Elsevier Procedia Computer Science, Vol 78,2016, Pages 291-297

[8] Alok Mishra, Ziadoon Otaiwi, *DevOps and software quality: a systematic mapping* Elsevier Computer Science Review, Vol 38, Nov 2020, 100308.

[9] Debois P., (2008), *Agile infrastructure and operations: how infra-gile is you?* Agile 2008 Conference, IEEE, Toronto, ON,Canada, ISBN: 978-0-7695-3321-6

[10] Khan AA, Shameem M. Multicriteria decision-making taxonomy for DevOps challenging factors using analytical hierarchy process. J Softw-Evol Proc. 2020; 32(10):11-13, e2263.

[11] Leite L, Rocha C, Kon F, Milojicic D, Meirelles P. (2019), *A survey of DevOps concepts and challenge,* ACM Computing Surveys (CSUR). 2019; 52(6):1-35

[12] Trihinas D, Tryfonos A, Dikaiakos MD, Pallis G (2018). *DevOps as a service: pushing the boundaries of microservice adoption*. IEEE Internet Comput;22(3):65-71

[13] Silva, M.A., Faustino, J.P., Pereira, R., da Silva, M.M.: Productivity gains of DevOps adoption in an IT team: a case study (2018)

[14] Stoneham, J., Thrasher, P., Potts, T., Mickman, H., DeArdo, C., Limonchelli, and T.A.: DevOps Case Studies: The Journey to Positive Business Outcomes. IT Revolution Press. Portland

Poonam, Research Scholar, pursuing PhD from the Department of Computer Science and Applications, Maharishi Dayanand University, Rohtak, Haryana under the supervision of Respected Dr. Pooja Mittal (Research Guide and Second Author). Author's Qualification is M.Phil. (CS), MCA. She had attended many National and International Conferences including Springer and IEEE and also published many research papers. She can be contacted at **email**: poonam.mehta20@gmail.com, **Orcid ID** - 0000-0001-7949-344X

Dr. Pooja Mittal obtained her Ph.D. degree from Maharshi Dayanand University. Her area of research and specialization include Data Mining, Data Warehousing, and Computer Science. She had published more than 50 research papers in renowned International and National Journals and attended more than 30 Conferences. Currently she is working as Assistant Professor in the Department of Computer Science & Applications, Maharishi Dayanand University, Rohtak (Haryana). She can be contacted at **email**: mpoojamdu@gmail.com, **Orcid ID** – 0000-0001-9746-6621