A Qualitative Assessment of Traditional and Agile Software Development Methodologies Together with DevOps Culture

Poonam Narang[1*] and Pooja Mittal[2]

[1]Research Scholar, [2]Assistant Professor, Department of Computer Science and Applications, MDU, Rohtak, Haryana, [1]poonam.mehta20@gmail.com, [2]mpoojamdu@gmail.com

Abstract - Aim of Software Engineering is to focus primarily on high Quality, well documented and within budget software, that too in a small pace of time. To achieve this target, software engineering propounds several software development frameworks that are broadly categorized into Traditional Methodologies, Agile Methods and DevOps, an emerging trend in software engineering. These methodologies cover different models viz. Waterfall, Prototype, Spiral, Iterative, Evolutionary, V Models, RAD, Scrum, Kanban, XP, Crystal, DSDM etc. But the existence of several different models or methodologies raises the need for their accurate selection for the successful delivery of software. Also agile development methodologies have emerged as a big alternative to traditional methods. As a whole, the target of every looming model is to neutralize the disparities of earlier models. This paper considers Waterfall, Prototype, Spiral, Iterative methods as representatives of Traditional methodologies along with Scrum, RAD, Kanban, and XP as representatives of agile. These representatives are compared analytically with one another followed by in-depth comparison with DevOps culture. Latest Google Trends and Stack Overflow Trends are also included to show interest over time for these models. Much empirical research has been conducted giving precedence to agile methods over traditional methodologies. However, considering the first waterfall methodology to recent agility in methods, still many impediments are found that make the task of appropriate methodology selection for software development very challenging for the IT industry. The purpose of underlying research is to provide a comparative study report of these representatives to target this particular challenge of the IT industry. The analysis and review carried out under this work will be useful for a divergent group of researchers/students to understand the course of action of different development methodologies including DevOps.

Keywords: Agile, DevOps, SDLC, Traditional Model.

I. INTRODUCTION

Different Methodologies and models exist that follow the step by step guide for software development in terms of life cycle (SDLC). Software or the System Development Life Cycle (SDLC) refers to the conceptual framework that clearly describes the schema for the software development. It defines all phases from feasibility study to the deployment, Operations, Maintenance and even obscurity of the software. These include many of the traditional models to agile development and presently trend moving towards continuity of development and operations through DevOps. Existing literature agrees on difficulties encountered in traditional models and accepts the requirement of more agility and continuity in software development. Thus DevOps, a fine blend of Development and Operations teams, shoots up as the vividly applied formula in software engineering. DevOps permits developers and operations to collaborate and work together more effectively and efficiently.

Study performed here covers three frameworks for methodologies and also discusses different models under these methodologies. Although SDLC models exists in great quantity, this research work analyses most widely followed models categorized and shown below -

Table1. SDLC Models Categorization under different Methodologies as under

Methodology	Development Model
Traditional	Waterfall, Prototype, Iterative, Spiral
Agile	RAD, Scrum, Kanban, XP
DevOps	DevOps

 DOI : 10.35291/2454-9150.2022.0174

As categorized in table (1), this research has been conducted to exhibit and reveal the challenges that occur in traditional models including agile methods. We also provide a tabular comparison with DevOps - an emerging phenomenon in software engineering that becomes the main contribution of this study. Also this review will be advantageous for students/researchers to understand the modus operandi of all development methodologies along with DevOps. As a part of this research, many existing papers have been studied that we have referred to in this paper also but none of those authors covered all these development models and that was my personal thought that at least our young buddy researchers could get all the data at one place. This work will also be useful for both academics and industries to understand the work on different dimensions of DevOps. As a part of future research, this extensive study can further be extended to work on DevOps applications from the development industry to software academics.

The remainder of this paper is structured as follows-

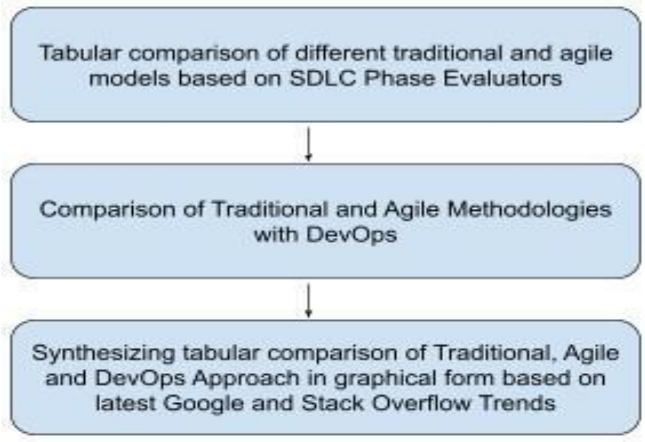

Figure1. Schematic Diagram for the complete comparison of different software development methodologies viz. traditional, agile and DevOps culture

Above figure (1), clearly depicts the flow of this study as after tabular comparison of traditional and agile methodologies individually based on performance evaluator parameters, generic comparison of Traditional vs agile and agile vs DevOps was made. Finally, all three methodologies were also compared graphically.

II. RELATED WORK

Application and utilization of Traditional and Agile Models including DevOps are reviewed from different existing recent research papers.

Nayan B. Ruparelia [1] accepts that the development models of software can be categorized as three broad groups viz. linear, iterative and a combination of linear and iterative models. This research paper confirms that SDLC Models exist in the big list but only the important or popular models viz. Waterfall, Spiral, Unified, Incrementing, Rapid Application Development, V and W were considered. The research gap of this paper lies in terms of inclusion of continuity in agile models. Marian STOICA et al [2] in their paper on Comparative analysis of traditional and Agile models abbreviated results as no impact of development model chosen, agreed on predisposal of errors in testing and validation processes. Thus the research put greater emphasis on testing and validation before entering into production. This research also confirms the inclusion of the client side for development and implementation of the project as per their specifications.

Ernest Mnkandla and Barry Dwolatzky [3] in their paper on Agile Methodologies confirm that agile methods provide a significant improvement on the control and management of the software development process. Their research paper also devised a mechanism that can be used by practitioners to select the most suitable agile methodology for a given software development project. The work also confirms that Literature gives evidence of improvement in areas like development of software that meets the user requirements, delivery of the product on time and within budget. Pulasthi Perera et. al. [4] and F. M. A. Rich et al [5] contributed a paper on Improving Quality through practicing DevOps conducts a study on how DevOps practice has impacted software quality. The secondary objective of this study was to find how to improve quality efficiently. In [5], authors also specified the way organizations are adopting DevOps for software development and improving software quality.

Authors of [5] also contributed a paper on mapping Study on Cooperation between Information System Development and Operations [6] covering detailed study for exploring DevOps. Authors also agreed upon no existence of adequate study of DevOps in scientific literature. This paper supports DevOps as a benefit for IS development and Operations performance. Lucy Ellen Lwakatare, Pasi Kuvaja and Markku Oivo [7], identified DevOps as an important aspect in academics and practitioner communities. Their study also investigates elements that characterize DevOps through literature survey and interviewing practitioners who are actively involved in the DevOps movement. Leah Riungu, Kalliosaari, Simp Makinen, Lucy ellenLwakatare, Juha Tihonen, Tomil Mannisto [8] conducted a qualitative multiple-case study and interviewed the representatives of three software development organizations in Finland. Authors observed the DevOps encouragement of collaboration between departments which boosts communications and employee welfare.

Mishra and Otaiwi in their work on mapping DevOps and Quality [9], mainly focuses their research in automation, culture, continuous delivery and fast feedback of DevOps.

 DOI : 10.35291/2454-9150.2022.0174

Their resulting study provides a better understanding of DevOps on software quality for both professionals and other researchers. Another work on DevOps published in [10] reveals different challenges experienced by software firms in DevOps adoption.

Above discussed and other existing survey or research work speaks individually about Traditional, Agile and DevOps methods. Also studies were there discussing comparisons of these models but the work in this paper performs a systematic study and moves gradually towards DevOps from Traditional through Agile. Parametric comparison tables are also given as theoretical evidence of practicing DevOps to encourage quality and speedy delivery of software.

Traditional Methodologies available for Software Development

The existing or most popularly followed traditional methodologies for the development of software inculcates many models like Waterfall, Prototype, Incremental, Iterative, and the V-Model etc. These methodologies proffer different phases of development to undergo during the Software Development Cycle. All of these models bestow many advantages during the development viz. easy system building process, reduction in failures and touching the customer's needs in actual.

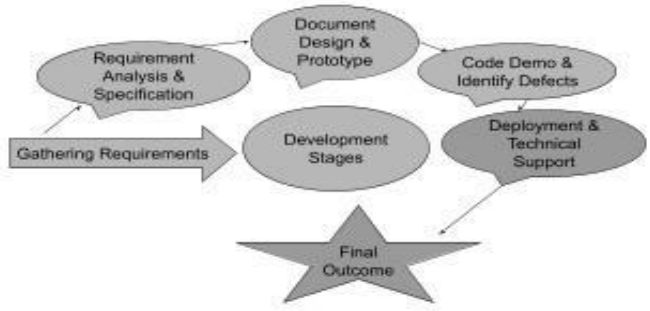

Figure2. Different Software Development Life Cycle Stages in Traditional Methodologies

Above figure (2), includes different phases of traditional software development methodologies. Maximum development models follow earlier defined phases/stages. All of the stages are explained beneath -

Requirement Excerption and Selection - This phase identifies customer requirements and accords their documentation on a very high level of abstraction. These gathered and refined requirements are fed as input to the next phase of design and implementation. After analysis of requirements, this phase ends up with the surety of meeting all system requirements.

Design - The design phase of the system works on specifying the features and operations to meet the expectations of the final product. The specifications document along with design outcomes are maintained collectively. All constraints of time, budget, technology, risks etc. are all considered and also discussed with associated teams and the customer to affirm the best design approach for the product.

Coding - Design phase completion calls upon the next goal of translating the system design into well structured, clear, understandable and simple language code that too in the best possible manner. After writing the complete code, many of the methods like code review and inspection, data flow analysis etc. are also followed for the purpose of verification and validation.

Testing - Scrupulous testing techniques are employed to check whether the designed and coded system meets the expected customer and system requirements. This phase includes many different types of testing including integration and system testing. Some definitions of testing from the literature consider testing as the process of executing a program with the intent of finding errors [11]. Whereas Hetzel, W in his complete guide to software testing [12] defines testing as the process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies specified requirements or to identify differences between expected and actual results. Royce, W. [13], on the similar platform states that testing can be any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Testing is the measurement of software quality.

But none of the above definitions authenticates error free software even after rigorous testing.

Implementation - This phase occurs when the majority of the program code is written and the project is ready to put into production. This is the actual doing phase and it is very important to maintain its impulse. At the end of this phase, the outcome is evaluated with the earlier created requirements and designs, whether the outcome is as per the specifications of the definition phases. Only after meeting full consistency, this phase is considered complete.

But meeting all the requirements specified in the defining phase is hardly ever possible as unexpected events or in-depth insight into a project raises the need of deviation from the original list of requirements. Now this could be the source of conflict but customers can appeal to agreements made during the definition phase.

Installation/Deployment - This phase, in the life cycle of software development, comes when the final software is released on the approval of development, testing and implementation teams. This phase finally handed over the

 DOI : 10.35291/2454-9150.2022.0174

software to the customer/client to get installed on their specific devices.

Operations and Maintenance - After the installation phase, the operations phase begins followed by the maintenance phase that starts when the customer/client needs any kind of update or modification in the installed product. At this time, users fine-tune the system as per their expectations or additional requirements. In short, maintenance phases can make changes to either hardware, software or the documentation part of the product with the specific intention of improving the product's final operational effectiveness.

Traditional Models for Software Development

Traditional methodologies are followed by many software development models. In this paper, Waterfall, Iterative, Prototype and Spiral Software Development Models are taken into consideration. These models are briefly discussed beneath.

Waterfall (also known as Traditional Model)

Waterfall or traditional model of software development is still used widely and will be around for a much longer period of time only because of the model's simplicity and convenience of usage. Different issues that come with this model are its less acceptance of change as to avoid the rework and software quality degradation.

Waterfall model for software Development follows steps similar to all mentioned under traditional methodologies as displayed in diagram 2 above.

This model takes the advantage of amputating the complete development life cycle into distinguished stages/phases that too are very easy to understand and to follow by the management team. These stages include engineering of requirements, choosing best design for software, design implementation in terms of programming code, testing of complete software, release and operations followed by the maintenance phase. Documentation and quality check processes are also run-through these phases.

Evolutionary Prototype Model for Software Development

Earlier software development life cycle models assume that customer requirements can be frozen in the beginning as they are consistent and unambiguous. But this assumption seems to be extremely rigid as requirements tend to change very frequently. In the Evolutionary prototyping development method, a prototype is first constructed by the development team and many different prototypes are built to be concise with customer feedback. This cycle moves on until the emergence of the final product.

This prototyping scheme differs from the rapid or throwaway prototyping in the way of requirement understanding and initial outcome of prototypes.

Iterative Model of Software Development

The iterative software development life cycle model initially focuses particularly on simplified implementation but followed progressively by a much broader and complex set of features for the inclusion in the final system.

The iterative model is better clarified with the following figure taken from professionalqa.com page for iterative model.

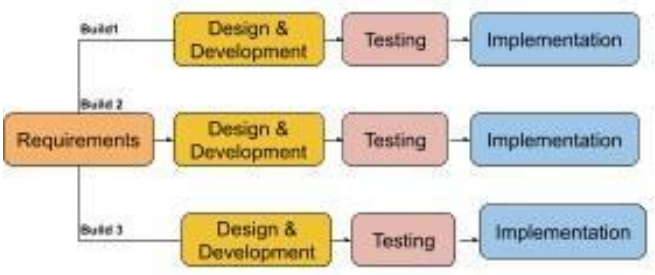

Figure3. Iterative Model for Software Development under traditional methodology

Software Development with the iterative model, shown in above figure (3), well suits the large size projects with their incomplete and unclear requirement specification. This approach permits the development of products based on their high priority requirements first and thereby also reduces development time up to greater extent. Besides these, iterative models are comparatively simple to use and understand that makes it a widely used development model.

Spiral

The spiral life cycle model came into existence to address challenges that occur in earlier development models such as coping with rapid change while assuring high quality and reliability at the same time. This model also addresses the need for rapid updating of incremental features or requirements with a minimum of rework.

A sketch of the spiral model is outlined in the figure below.

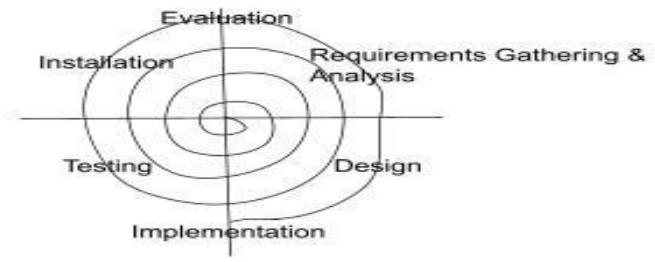

Figure4. Spiral Model for Software Development falls under traditional methodology

 DOI : 10.35291/2454-9150.2022.0174

Different loops of the spiral model, shown above in figure (4), are called a phase of the development process. Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants can be classified as Determination of objectives and identifying alternative solutions, Identifying and resolving risks, developing the next version of the product, and in the last reviewing and planning for the next phase.

The number of phases that are required to develop a software product depends upon the number of risks identified and can be varied with the project.

III. AGILE METHODOLOGIES

Agile development methodologies break the traditional software development process into much smaller size blocks that could easily be managed. Agile approach focuses on people, results and rapid delivery of the product. It rotates around well defined and well organized tasks, short delivery times and at the same time adaptive planning to accept the changes at any point during the development process.

Agile relinquishes the risk of spending months or years on a process that ultimately fails just because of some small mistake in early phases. Instead, it relies on its team of employees and developers to communicate directly with their clients to get a clear understanding of goals and thus provide alternative solutions in a fast and incremental way.

Agile can be better understood by the diagram below drawn with the help of reqtest.com page about agile methodologies.

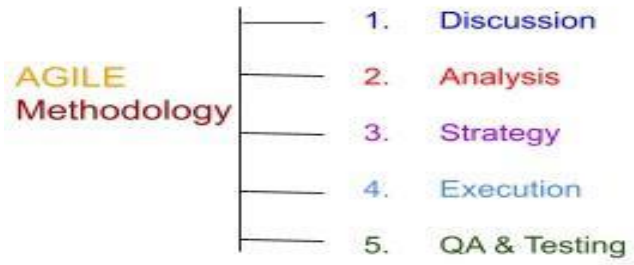

Figure5. Agile Methodology for Software Development

Agile methodology displayed in above figure (5), clearly indicates different phases of development. These are – Plan/ Discussion, Detailed analysis followed by making strategy/ design for solution, execution and Testing with proper quality assurance. Some most commonly used and popular examples of agile methodologies are- Rapid Application Development (RAD), Scrum , Feature Driven Development (FDD), Crystal and Lean Software Development (LSD), Kanban, eXtreme Programming (XP) etc. This research work discusses RAD, Scrum, Kanban and XP development models.

RAD (Rapid Application Development)

Rapid Application Development is the development model that is designed specifically to deliver quality results at a rapid pace of time as compared to traditional methods. Its main emphasis is on development rather than planning. RAD models follow the approach of time boxing that permits multiple cycles development at the same time.

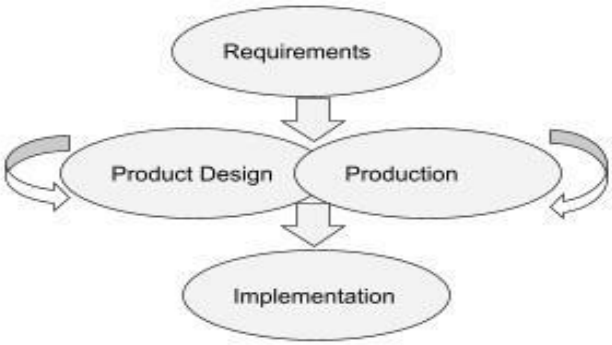

Figure6. RAD Model for Software Development under Agile methodology

Above figure (6), also shows that after the completion of each module, feedback is also acquired from the clients and the management team. The RAD model suits for each size of project be it small, medium or large scale. The only requirement of the RAD model is to divide the project into smaller manageable chunks.

Scrum

Agile's other way or approach to manage complex projects is by scrum software development that follows an incremental approach. In the Scrum model, the project progresses through a series of sprints where sprint is a planned amount of work that the team is to complete within the specified time box of generally 2-4 weeks iteration of a continuous development cycle. For this purpose, a sprint backlog is also maintained that contains all the tasks to be performed during the sprint cycle. During the sprint cycle, daily scrum meetings are also conducted for the team members including Scrum Master and the client. This daily meeting is time boxed to 10-15 minutes in which sprint progress along with the next plan is discussed by all members.

 DOI : 10.35291/2454-9150.2022.0174

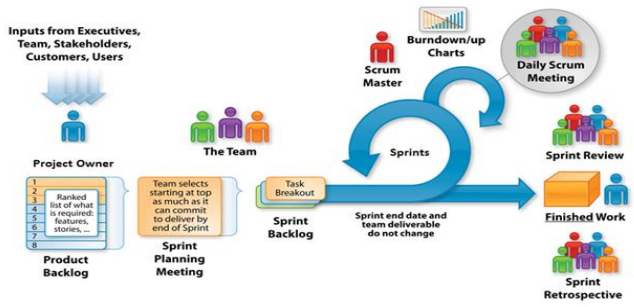

Figure7. Screenshot of Scrum Model for Software Development under Agile Methods [14]

Thus we can see the Scrum model, in above figure (7), as a methodology to synchronize the team members work while discussing sprint work. The Scrum is also suited for all sizes of projects small, medium or large scale.

Kanban

Kanban, a popular agile model, requires real-time communication of work progress along with its full work transparency on a Kanban board. Work items represented visually on the board allows team members to see each and every progress of work. The major benefits of using Kanban are its better understanding of the development process and good effort to cover all customers' requirements.

Figure8. Kanban Model screenshot for writing notes during Software Development [15]

Kanban is a Delivery model with no fixed scope or deadline as depicted in above figure (8). Work is getting prioritized continuously and work items get delivered as soon as they are ready for production, hence no fixed date of delivery. Key ingredients in the success of such a model are close and continuous collaboration within the team: Production Owner, BA, Developer and Tester. New requirements are captured in a state called Backlog/Icebox and the Team along with the Product Owner discusses the priority of those work items daily, and as soon as work on one requirement is completed, the team picks the next items from the priority list to deliver.

eXtreme Programming (XP)

eXtreme Programming (XP) is one of the most important software development models that come under agile methodologies. It improves software quality and response time to client's requirements. The term extreme comes from the fact that the underlying model takes the best practices that have worked very well in the past during the software development to extreme levels.

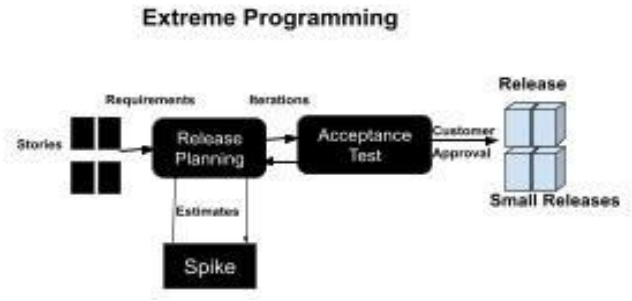

Figure9. Different iterations for XP Model of Software Development under Agile Methodology

eXtreme Programming model, shown in above figure (9), is based primarily on frequent iterations for implementing user stories where user stories are very simple and informal statements from the customers about the needed functionalities in the product. On the basis of these stories, the project team proposes their common vision called metaphors about the working of the system. In view of the metaphors, the development team may construct spikes that refer to simple programs or also called solutions in terms of prototype.

DevOps or Development + Operations: An Introduction

DevOps approach is the most recent addition to the software development research area. A strong practitioner-driven movement supporting the idea of using DevOps in software engineering has emerged. It is a set of practices that work together to automate and integrate the processes between development and Operations teams, so that they can build, test, and release software much faster and with better reliability. The term DevOps was formed by combining the words "development" and "operations" that bridges the gap between the development and operation teams, which in history functioned in siloes.

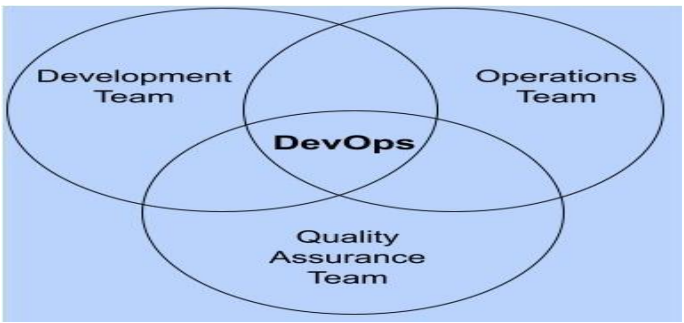

Figure10. DevOps = Development + Operations (A fine blend of different teams)

 DOI : 10.35291/2454-9150.2022.0174

DevOps, as shown in above figure (10), is a fine blend of development and operations teams in terms of better communication, continuous testing, integration and deployment with the use of different automation tools at each and every stage of software development. DevOps combines agile, git, automation and much more efficiency to get faster and quality delivery.

DevOps becomes very easy to implement when we have the surety of adoption of changes during the development cycle. DevOps methodology thrusts for the speedy delivery of the product for which it adopts many automation tools viz. Jira, Git, BitBucket, Selenium, Docker etc. DevOps is based on the principle of 5 C's. These are

- Continuous Integration
- Continuous Testing
- Continuous Deployment
- Continuous Delivery
- Continuous Monitoring

DevOps develops Integrates, Test, and Deploy, Deliver and even monitor the software product continuously means everything goes parallel that makes the development and operation teams work in collaboration to deliver a successful product that too in a very short span of time. DevOps is easy when you know your organization can adopt changes easily and when you have the right attitude to make DevOps come true in your organization.

Comparative Analysis of different Traditional and Agile Methodologies

Traditional methodologies to software development strictly require well defined and well planned tasks that too to be completed in a stipulated time period to deliver the within budget and successful projects. These methods completely depend upon the detailed perusal of the requirements along with a careful planning from the early cycles of the development. As any change afterwards or during the development requires a rigorous amount of rework and change control management checks are also needed at the same moment. Traditional methods, on the other hand, emphasize heavy documentation, so are best suited for quite large sized projects.

Agile methodologies are based on workable methods that accept the beginning of development much before any project planning. These methods work with the only requirement of clear and crisp understanding of all tasks or actions that are to be performed during the development. These models follow the adaptability of any new change in the requirements later in the project. The product is tested, therefore, much frequently only for the sake of nullifying the risk factor effect associated with the occurrence of faults. The approving points of Agile methodologies lie in their very less documentation work, a continuous interaction with their clients and also good team alliance. This can also be concluded that these methods are best suited for the longer projects that are easy to be modularized.

A phase and parameter-based comparison of these traditional models viz. Waterfall, Prototype, Iterative and Spiral along with different agile models viz. Rapid Application Development, Scrum, Kanban, eXtreme Programming are detailed in the following table.

Table2. Traditional Models Comparison with respect to parametric evaluative features of different phases/stages based on existing literature, research or survey of many renowned researchers [16] – [23]

(* table shown on the last page 13-14 of this manuscript)

A detailed study of different development models under traditional and agile methodologies for their respective advantages , differences, and drawbacks permits us to make a generalized comparison between the two based on different parameters. This generic comparative study not only helps for the selection of appropriate development models for the underlying problems but also ensures the successful implementation.

Traditional Methodologies and Agile Methods compared with DevOps

Followed by detailed analytical comparison of traditional and agile methods, these are again compared with DevOps. Traditional methods suitable for large and freeze requirements differ from DevOps with respect to industry latest trends and interest over time. On the other hand, DevOps and agile methodologies, contrasts from one another in terms of their basic footing principle. Agile methods are based on the principle of agility in development whereas DevOps bring agility in development including operations as depicted in the following diagram –

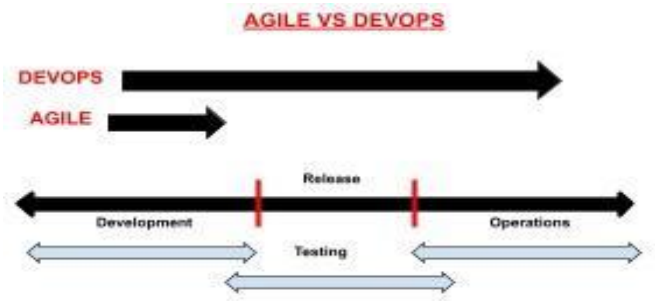

Figure11. Agile methods and DevOps consideration of different approach

 DOI : 10.35291/2454-9150.2022.0174

Differences among these development methodologies other than the shown above in figure (11), are summarized in the following table.

Table3. Traditional and Agile Methods comparison with DevOps based on different evaluative parameters [24] – [27]

(* table shown on the last page no 15 of this manuscript)

Other differences can also be viewed in Agile and DevOps in terms of continuity in the application of agility in development. Agile models overweigh the development of testing, release and operations. DevOps, on the other hand, considers testing and operations of a product equally to the development as we observe in the figure below.

Further comparison of traditional and agile methods including DevOps is shown in the following figure.

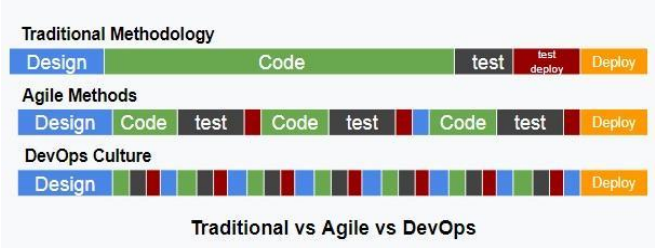

Figure12. Traditional, Agile and DevOps Comparison with respect to continuous approach

Above comparison, taken from many renowned researchers' work available on the internet, confirms that all these are not seen to be incompatible, leading to the existence of all three, but still the continuity approach of DevOps leads to future possibility of **DevOps based Software Engineering (DOSE)**. DOSE includes automation at each and every phase of software development to make it more reliable, faster and quality.

Interest over Time of different Traditional, Agile methods and DevOps based on Latest trends in graphical form

Over the decade or two, the mindshare of the developer has transitioned from traditional development methodologies to agile methods as of their usage of variety models with well defined purposes and distinguished approach to development. Other than these, agile methods exhibit very slight cost of rework as compared to traditional methodologies. Despite these differences, we can observe in the following graph about interest in Waterfall, Iterative, Spiral and Prototype methods according to the latest trends.

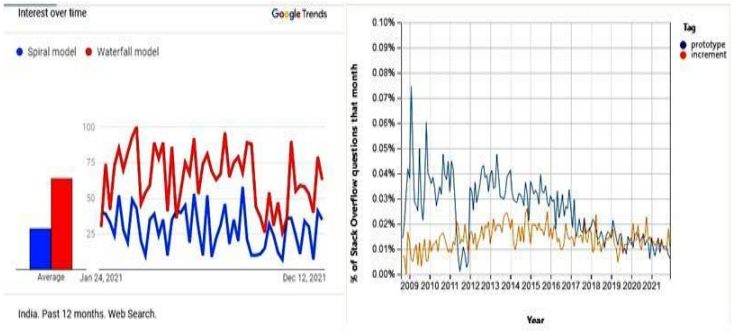

Figure.13. Traditional methodologies interest over time of past 12 months as per Google trends and over period of 7-8 years according to Stack Overflow Trends as on 18 Jan 2022 [28] [29]

As clearly depicted in the above figure (13), traditional methods are still in demand because of their simplicity, ease of understanding and easy applicability at the same time.

Agile methods are also in trend according to Google and Stack Overflow trends as described beneath –

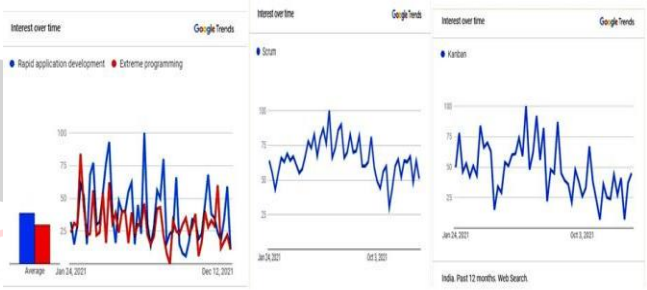

Figure14. RAD, eXtreme Programming, Scrum and Kanban Agile methods interest over time according to Google Trends over the past 12 months [30] [31] [32]

All models that fall under agile methods are in usage with almost equal interest by IT industrialists as depicted in the above figure (14) by Google Trends report. These trends accept that Traditional and Agile methods are followed as per the underlying project requirements. In a similar way, we can also confirm the popularity of DevOps according to both trends as shown below –

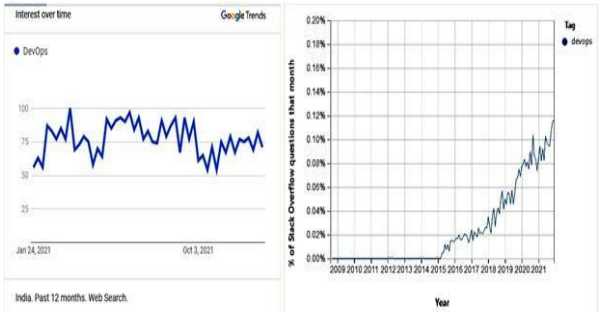

Figure15. DevOps interest over time of past 12 months as per Google trends and over period of 7-8 years according to Stack Overflow [33] [34]

 DOI : 10.35291/2454-9150.2022.0174

DevOps also gains popularity according to the above trends shown in figure (15). During Covid-19 Pandemic, DevOps became the latest buzzword in the IT industry because of its on time, within budget and quality software delivery.

IV. CONCLUSION AND DISCUSSION

The ample increase in the number of software development methodologies or models makes the selection of a suitable approach for a particular project all-important. This extensive research of different methodologies - Traditional, Agile along with DevOps will be a great help to the developers as the motive behind such a paper is the availability of several software development methodologies for which effective selection becomes a very difficult and time consuming task.

The underlying study makes a comparative analysis between different Traditional and Agile Methods followed by a tabular comparison between Traditional vs Agile vs DevOps based on different evaluative parameters. Though traditional software development methodologies are quite older but still possess such qualities that even depending upon the requirements, they are very much used in the development zone. Thus, waterfall is simple to implement, Agile with little documentation, and DevOps for the latest approach features makes them all equally important even till date.

It can be concluded that any model can be opted as per the requirement by the developer. Every model has its own strength. This research concludes that traditional methods are very easy and simple to follow with the restriction of not frequently changing requirements. Agile methods take advantages over traditional methods like very little documentation, fast release, and small cycles. Conclusion of this comparative study considers DevOps to be the latest approach for development of software to cover the speed and quality oriented concerns of organizations.

V. FUTURE WORK

The work put here can be used by a divergent group of researchers or students to get an in -depth understanding of different development models either traditional, agile or DevOps. This research work can be further extended to consider remaining models of Traditional or Agile methodologies as a part of future research. Further research work can also be done on working principles of DevOps along with its automation tools.

"Compliance with Ethical Standards"

"**Funding** - This study was not funded by any authority."

"**Conflict of Interest:** The authors declare that they have no conflict of interest."

REFERENCES

[1] Nayan B. Ruparelia., "*Software Development Lifecycle Models. ACM SIGSOFT Software Engineering Notes"*, May, Volume 35, Number 3, pg 8 (2010).

[2] Marian STOICA, Marinela MIRCEA, Bogdan GHILIC-MICU., *"Software Development: Agile vs. Traditional. Informatica Economică",* vol. 17, no. 4/2013.

[3] Ernest Mnkandla, Barry Dwolatzky, "*A Selection Framework for Agile Methodologies"*, International Conference On Extreme Programming And Agile Processes In Software Engineering, Xp 2004, Lncs, Vol. 3092, Pp. 319-320. IEEE (2018).

[4] Pulasthi Perera, Roshali Silva, Indika Perera.,"*Improve Software Quality Through Practicing DevOps"* ,2017 Seventeenth International Conference On Advances In Ict For Emerging Regions (Icter), Eissn: 2472-7598. IEEE (2018).

[5] F.M.A. Rich, C.Amrit,M. Daneva, "*A Qualitative study of DevOps usage in Practice",* Journal of Software: Evolution and Process, Wiley Online Library, June 2017

[6] Floris Erich, Chintan Amrit, and Maya Daneva: A Mapping Study on Cooperation between Information System Development and Operations. Conference Paper, Part of the Lecture notes in Computer Science book series(LNCS, volume 8892), PROFES 2014, pp 277-280

[7] Lucy Ellen Lwakatare, Pasi Kuvaja and Markku Oivo: Dimensions of DevOps. Conference Paper, Part of the Lecture notes in Business Information Processing book series (LNBIP, volume 212), XP 2015, pp 212-217.

[8] Leah Riungu, Kalliosaari, Simp Makinen, Lucy ellenLwakatare, Juha Tihonen, Tomil Mannisto, *"DevOps Adoption Benefits and Challenges in Practice: A Case study",* PROFES 2016, pp 590-597, Springer International Conference on Product-focused Software Process Improvement.

[9] Alok Mishra, Ziadoon Otiawi, "*DevOps and software quality: A Systematic Mapping"*, Elsevier Publications, Computer Science Review, vol 38, Nov 2020,100308

[10] JAVMK Jayakody, WMJI Wijayanayake, "*Challenges for adopting DevOps in information technology projects"*, published in proc. 2021 International Research Conference on Smart Computing and Systems Engineering (SCSE), 16 Sep 2021, IEEE Xplore.

[11] http://barbie.uta.edu/~mehra/Book1_The%20Art%20of%20Software%20Testing.pdf, last accessed 19/05/2022

[12] https://condor.depaul.edu/sjost/hci430/documents/testing/software-testing.htm,,last accesses 30/04/2021 ANSI/IEEE Standard 729, 1983

[13] https://condor.depaul.edu/sjost/hci430/documents/testing/software-testing.htm, last accessed 30/04/2021, Hetzel, W., The

 DOI : 10.35291/2454-9150.2022.0174

Complete Guide to Software Testing, QED Information Sciences Inc., 1984

[14] https://laptrinhx.com/introduction-to-agile-project-management-with-scrum-2904466361, last accessed 19/05/2022/

[15] https://www.istockphoto.com/photos/daily-scrum, last accessed 19/05/2022

[16] Dima, A. M., & Maassen, M. A. "*From Waterfall to Agile software: Development models in the IT sector, 2006 to 2018. Impacts on company management*", Journal of International Studies, 11(2), 315-326. doi:10.14254/2071- 8330.2018/11-2/21 (2018)

[17] Krishna, S.T, Sreekanth, S., Perumal, K., & Reddy, K.R.K., "*Explore 10 different types of software development process model*", International Journal of Computer Science and Information Technologies, 3(4), 4580-4584 (2012).

[18] Chandra, V., " *Comparison between Various Software Development Methodologies*", International Journal of Computer Applications, 131(9), 7-10, Retrieved 18 Jan 2022, from: https://www.ijcaonline.org/research/volume131/number9/chandra-2015-ijca-907294.pdf (2015)

[19] B.W. Boehm, "*A Spiral Model for Software Development and Enhancement*", IEEE, IEEE Computer Society, vol. 21, issue 5, May 1988, pp. 61 – 72.

[20] David J. Anderson,Giulio Concas,Maria Ilaria Lunesu, Michele Marchesi, Hongyu Zhang,"A Comparative Study of Scrum and Kanban approaches on a Real Case Study Using simulation", International Conference on Agile software Development,(LNBIP, Vol 111), Springer Link (XP 2012)

[21] Stober, T., & Hansmann, U., "*Agile Software Development: Best Practices for Large Software Development Projects*", Berlin, Springer Verlag (2010)

[22] McHugh, M., Cawley, O., McCaffery, F., Richardson, I., & Wang, X., "*An Agile V-Model for Medical Device Software Development to Overcome the Challenges with Plan-Driven Software Development Lifecycles"*, IEEE 5th International Workshop on SEHC, (2013)

[23] Schaefer, A., Reichenbach, M., and Fey, D., "*Continuous Integration and Automation for DevOps*", Lecture Notes in Electrical Engineering, pp. 345-358, 2012.

[24] Prashant Agrawal, Neelam Rawat, "DevOps, A New Approach to Cloud Development and Testing", International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), IEEE, 2019

[25] Boyuan Chen, "*Improving the software logging Practices in DevOps*", IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (2019)

[26] J. Humble and J. Molesky, "*Why enterprises must adopt devops to enable continuous delivery*", Cutter IT Journal, vol. 24, no. 8, pp. 6, 2011

[27] M. Virmani, "*Understanding devops bridging the gap from continuous integration to continuous delivery*", in 5th International Conference on Innovative Computing Technology (INTECH), 2015

[28] https://trends.google.com/trends/explore?geo=IN&q=%2Fm%2F0136fk,%2Fm%2F0867l, last accessed 19/05/2022

[29] https://insights.stackoverflow.com/trends?tags=prototype%2Cincrement, last accessed 19/05/2022

[30] https://trends.google.com/trends/explore?geo=IN&q=%2Fm%2F01jtfr,%2Fm%2F02t2n, last accessed 19/05/2022

[31] https://trends.google.com/trends/explore?geo=IN&q=%2Fm%2F0ck_p8, last accessed 19/01/2022

[32] https://trends.google.com/trends/explore?geo=IN&q=%2Fm%2F0dgq419, last accessed 19/01/2022

[33] https://trends.google.com/trends/explore?geo=IN&q=%2Fm%2F0c3tq11, last accessed 19/05/2022

[34] https://insights.stackoverflow.com/trends?tags=devops, last accessed 19/05/2022

 DOI : 10.35291/2454-9150.2022.0174

Table2. Traditional Models Comparison with respect to parametric evaluative features of different phases/stages based on existing literature, research or survey of many renowned researchers [16] – [23]

Methodology →		Traditional Models				Agile Methods			
Phases	**Features**	**Waterfall**	**Prototype**	**Iterative**	**Spiral**	**RAD**	**Scrum**	**Kanban**	**XP**
Requirement Elicitation and Analysis	Requirement gathering Requirement Analysis Requirement Change Stakeholder Involved	Freeze in Beginning Clear and Crisp Not feasible Beginning [16]	Throughout the phases Not very clear Yes High involvement of user	Throughout the phases Well Understood Yes Involvement in beginning only	Throughout the phases Well Understood Not Feasible Only in initial phases	Time Frame released Easy to understand Can be done In beginning phases only	Requirements change v frequently [16] Easy Understanding Very easy High involvement [20]	Requirements change v frequently Clear and Crisp Understanding Very easily High Level of Involvement	Change v frequently Difficult to understand Not Easy Only in beginning
Software Planning	Primary Concern Project Cost Resources & Cost Control Requirement of Expertise	Product Assurance Very Low Yes Highly Required	Speedy Delivery High No Not rigid requirement	Speedy Delivery High Yes Yes	Product Assurance Very Expensive Yes Highly Required	Fast Delivery of Product Very Low Yes Not as much	Progress and Budget As Estimated Sometimes over budget High Requirement	Customer Satisfaction Very High Yes up to some extent Very High Requirement	Product Assurance Very High Not at all Very high requirement
Product Design	Simplicity Overlapping Phases Flexible Documentation Expertise Required	Very Simple No Very Rigid [17] Must Required Yes	Simple Yes Highly [18] Not necessary Not as much	Intermediary Yes Very Flexible[18] Necessary Yes	Demanding Yes Highly Required Must Required	Much simpler Not at all Very much [17] Very less Less expertise required	Very Simple Yes Yes Less Yes	Very Complex Model Very Much Highly flexible Required Yes	Bit Complex Not at all Yes Required Up to some level

 DOI : 10.35291/2454-9150.2022.0174

Implementation	Time entailed Risk Anatomy Risk involvement Prominence on Success assurance	Very Long Yes High Requirements & Heavy Documentation Very less	Short No Low Design Good	Varies with requirements Yes Low Documentation Low	Long Yes [19] Very Low Documentation High	Much lesser [17] Very less Least feasibility Technical Strength of Product Very High	Much less Only at implicit level Not high Speedy Delivery High	Very Less Done rigorously Less prospects Requirement based Design Much higher	Very Short Not Considered Feasibility of Risks Fast Development Not much high
Integration and Testing	Testing Occurs Cost of Rework	After end of Coding Very High	After end of each iterative prototype model Not much High	After end of each iteration High	After end of Design phase Very High	After each phase Very less	After Coding phase Very High	Continuous Testing Approach Varies with project	Continuous testing with coding phase High
Deployment	Build Release Working Model Availability Product Customization User Control over Product	Not Possible End of Life Cycle Not Realizable Very Less [16]	After every Phase End of each Iteration Realizable High Control	After every Iteration End of each Iteration Very Much Realizable Much Control	Not Possible End of each Iteration Realizable Yes	In phases After each iteration Possible Yes	In Phases After each iteration Possible Very Much	As and when required Relies on JIT approach Possible Very Much [20]	In Phases After each iteration Possible Yes [16]
Maintenance	Emphasis on User Interface Maintainability Reusable	Very little Not Required Very Less	Very Critical Easy to maintain Can't reuse	As per User requirements Maintainable Component reused	Very Crucial Needs high maintenance Yes	Minimal Maintainable Yes	Very less Easy to maintain Can be reused	Less Easily maintainable Yes	Minimal Easy maintenance Yes

 DOI : 10.35291/2454-9150.2022.0174

Table3. Parametric evaluation based comparison of Traditional Methodologies and Agile Models with DevOps in support of existing literature review or survey [24]-[27]

Parameter	Traditional Methods	Agile Methods	DevOps
Basic Development Approach	Design begins after requirement Freeze	Heavily relies on speedy delivery of product	Speedy and quality delivery of product [26]
Feasibility of Change	Not possible [16]	Possible at any stage	Possible at any stage [25]
Development Oriented around	Process	People and Product	People and Product
Documentation Required	Heavy documentation	Very Less Documentation	Very light documentation required
Project Size	Big Size	Medium Size	Medium to small size
Collaboration among teams	Not at all	At development Level	High level of collaboration
Quality	Very Low	Improved Quality [18]	High level of Quality [27]
Risk Involvement	Increase with project progress	Decreases with progress	Decreases with progress of project
Customer Involvement	Only till requirement freeze	Very Frequent- after every sprint	Continuous involvement throughout the project
Automation	At very Low level	At varied level	High level of automation

 DOI : 10.35291/2454-9150.2022.0174