International Journal of Computer Applications (0975 – 8887)
Volume 178 – No. 53, September 2019

Software Development Methods – Properties and Advances

Syed Hamid Hasan
Department of Information Systems
Faculty of Computing and Information Technology
King Abdulaziz University, Jeddah, Saudi Arabia

Usman Ali Khan
Department of Information Systems
Faculty of Computing and Information Technology
King Abdulaziz University, Jeddah, Saudi Arabia

ABSTRACT
It has taken a period of time for evolution of the field of software development. In order to promote its effectiveness and efficiency propositions of various models and methods has been done. In this paper we are going to discuss the various models and methods, while identifying the latest industry trends and also discuss their impacts..

Keywords
Methods of Software development, Agile & Life cycle method, software testing.

1. INTRODUCTION
With the increasing demand of software in today's world, the way these software's are developed is also becoming critical. Researchers, practitioners, students, educators of software development are all focusing on this topic. The software development process has come a long way steadily evolving over the past 50 years from its humble beginnings. We have seen propositions of various models and methods (agile methods and life cycle model etc.) for software development to make it more effective and efficient. Life cycle model are the most used currently for software development specifically for larger software while agile methods are also picking up gradually. This paper makes an attempt to review the various methodologies of software development, while highlighting the latest trends and also discusses the effects of these models and methods on software development field. This would prove useful for researchers, practitioners, students and educators of software development.

2. MODELS AND METHODS OF SOFTWARE DEVELOPMENT
In this section various software development methods & models, like agile methods and analysis-coding & life cycle model have been reviewed. We have identified the strength and weakness for each model or method. It is noteworthy that none of them are "mutually exclusive" [1], in fact they are very commonly used in conjunction with each other specially when the software to be developed is a real time system or is integrated, complex and large.

2.1 The Waterfall Methodology
In Waterfall Models we come across a flow that is linearly sequential. We see progress steadily flowing downward (similar to a waterfall) through software implementation phases (Figure 1). It requires finishing of a preceding phase for the initiation of the subsequent phase. It means one phase in the process of development process cannot begin without the previous phase being completed. This approach doesn't provide methodology for going back to preceding phases for handling requests for change. This approach is considered to be the oldest approach along with being most commonly recognized approach in development of software.

A. Gathering & Documenting requirement.
B. Designing
C. Coding & Unit testing
D. Testing System
E. UAT(User Acceptance Testing)
F. Fixing issues reported
G. Delivering finished products

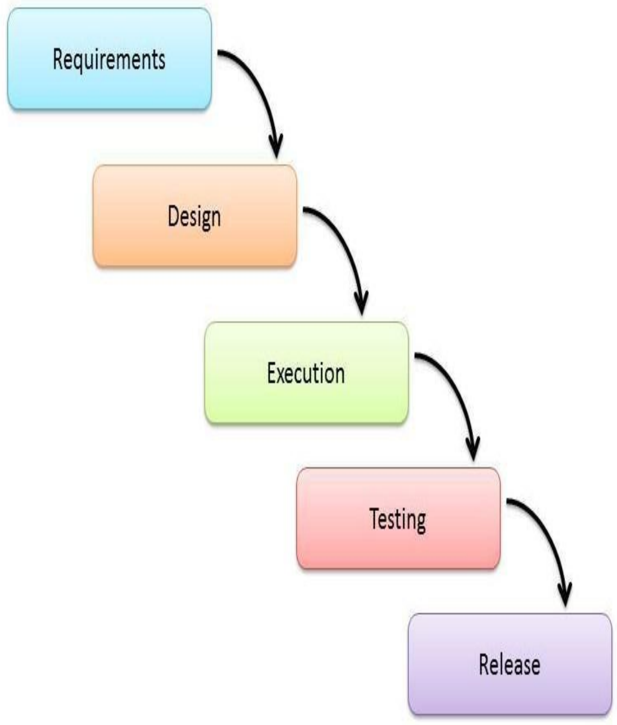

Fig 1: Waterfall Model

In projects religiously following Waterfall approach of development, every step mentioned represent separate stages in the software developmental process, & every stage must generally finish before beginning of the following one. Typically, there exists stage gates between each stages; e.g., requirement should be gone over and agreed upon by clients prior to designing phase beginning.

Like every other method, this approach has it's advantages and disadvantages [2]. Some of the advantages are listed below:

- There is an agreement on deliverables in initial phases of the developmental process between Client & developers. It removes ambiguity from planning & designing phases.
- We can easily measure progress at any point of time since, we already know the complete scope of work in the beginning of the project.
- Through the entire lifecycle of the project, team members can be engaged in the project or other activities based on which phase of the project is active and if they are required to participate in the specific phase, e.g. while the developers code, the business analyst could be learning about & documenting on what are the steps that need to be carried out. While the testing team can be preparing scripts for testing based on requirement documents.
- The customer does not need to be present for the project after the requirement is gathered apart for schedules status updates, approvals and progress review.
- Since designing is finished in earlier phases of development lifecycles, the approach is quite suitable for projects that require multiple component designing (maybe simultaneous) for integrating with systems that are external.
- Lastly, this approach enables software designs to be completer and more accurate, with its basis on more comprehensive understandings of the entire set of deliverables. It results in software designs that are better and less likely to be suffering from "piecemeal effect," which is a phenomenon in development that occurs when code snippets are defined & afterward added in applications that are suited for them or not.

Some of the disadvantages that the Waterfall approach presents are listed below:

- The primary area that mostly falls short of the optimum level is the requirement gathering effectiveness. To gather & document requirement so that they are meaningful for customers is frequently the hardest phase in software development, based on our observations. Clients are usually intimidated with detail & specifics, presented to them initially during the projects, while the approach warrants minutest levels of detail. Additionally, clients aren't normally capable of visualizing applications based on requirement documents. We can help with mockups & wireframes, but it is undeniably quite difficult for clients or end users for to imagine the end product based on the written documents and other elements of requirement gathering phases.
- One more disadvantage that the Waterfall approach of development poses is dissatisfied customers with end results when the project is finally completed. Since all the deliverables have their basis on the requirement document the customer may not really have an exact ide of the end product and might require certain changes which might be too costly/difficult in implementing.
-

2.2 Agile

This is a team based & iterative approach of software development. The emphasis of this approach on rapidly delivering applications with components that are completely functional. Instead of creating schedules & tasks, the duration of the projects are split or "time-boxed" into stages knows as "sprints." There is a specified duration for every sprint (mostly in weeks) and associated deliverables are decided, at the beginning of every sprint. Priority of the deliverables is done based on their value to business which the customer defines. In case the deliverables of a particular sprint is not completed, activities are re-planned and prioritized . The miss in attaining completion of the sprint is used as a lesson for planning of the forthcoming sprint.

Post completion of work, it's evaluated and reviewed by the client & project teams, via end of sprint & daily build demo sessions. Agile approach requires extensive level of involvement from the clients for the entire duration of the project which is intensified at time of reviews.

A few of the positives that Agile approach presents are:

- Clients have regular & early opportunity for seeing work deliveries, and making changes & decisions through the entire duration of the project development.
- Clients possess high sense of control when they work directly & extensively with project teams through the project entire duration of the project development.
- When specific applications have time to market as a bigger concern instead of full feature release that was initial agreed at launch, this approach is capable of quickly producing basic versions working software that can be enhanced in later iteration of the project.
- The focus of development is more on the user, possibly because of extra & frequent interaction by the client.

The disadvantages of the Agile approach are:

- Even though higher degrees of client involvements could be great for the project, but could be problematic for clients who don't have inclination or bandwidth to participate in the developmental efforts so extensively.
- This approach is best suited for project & teams where members can be dedicated to the projects and not hopping between multiple projects.
- Since Agile is focused upon time-boxed deliveries & regularly reprioritizing, possibilities are some delivery timelines may be missed. Additionally forthcoming sprints (post what were initially planned) could be required, increasing cost of the project. Additionally, client engrossment usually could lead to request for additional feature for the entire duration of the project. This would again have additional cost & time implications.
- Working relationships are really close in Agile approaches. They are easily manageable if team members have the same locations are close by for regular physical interactions. This may not be

possible always. Technology does offer tools like collaboration application and video & voice conferencing tools but they may not be as effective as physical presence.

- Agile development's iterative nature could result in recurrent refactorization when full system scopes of are not taken into consideration during the initial design & architecture. If this refactoring need is ignored, then overall system quality could be compromised or reduced. It is more noticeable when implementations are larger-scaled, or where there extensive level of integration within the systems.

3. MODELS BASED ON LIFE CYCLE

Eventually, models based on Life Cycle or the Life-Cycle Models were introduced that were aimed at bringing order and control into the process of software development. This model divided the process of software development into defined phases such as Analyzing, Designing, Coding, Testing and Implementing [3]. On the basis of the workflows involved the life cycle models may be further classified into Iterative (like Spiral) model, Progressive (like Phased) model and Sequential(like PV, traditional V and waterfall) Models (Figure 2, Figure 3).

3.1 Iterative Model

This model is designed for overcoming waterfall model drawbacks. Starting with initial planning, ending with deployment the process has interactive cycles in the middle. The elementary idea of this methodology is developing systems via iterative cycles and carried out in incremental time portions, this allows software developing team in taking advantages of learnings arrived during previous developmental cycles and older system versions. It is a combination of mini waterfall or V-Shaped models (Figure 6).

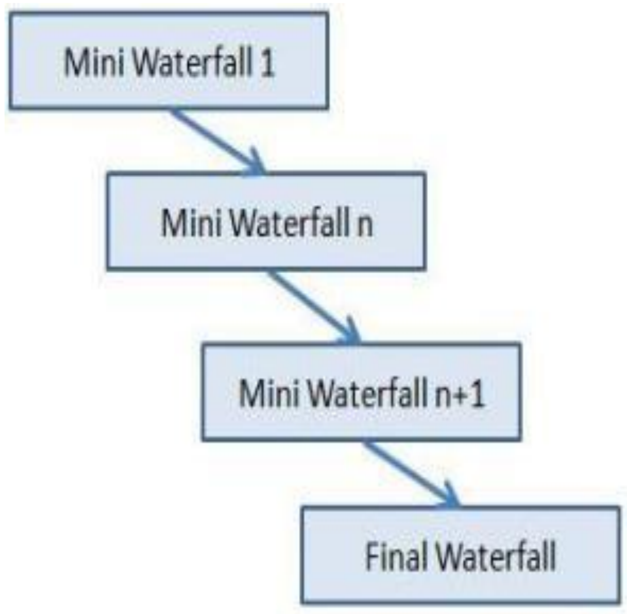

Fig 2: Progressive Model

It's utilized for applications that are shrink-wrap & hefty systems that have smaller segments or built-in phases. It can also be utilized for systems that have divided component, like ERP systems. Where the beginning could be with budget modules in initial iterations which can be progressed into inventory modules & so on.

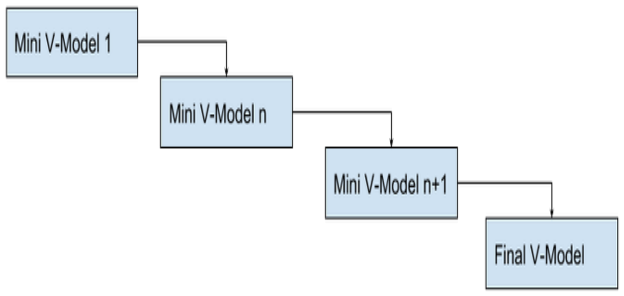

Fig 3: Progressive V-Model

Positive aspects	Negative aspects
• Producing value for business in early phases of development cycle. • Scarce resources are better used by properly incrementing definitions. • Option for accommodating change request is available between increment cycles. • Focus is on value for client more as opposed to approaches that re linear. • Issues & changes in the project detected earlier.	• Heavy documentation required. • Following of processes set quite restrictive. • Increments are defined on the basis of feature dependencies and function. • Client involvement is lot more in comparison to linear approached methods. • There could be a problem in partitioning of features and functions. • There could be issues in integration of modules among iterations if not properly considered during the project planning and development phases.

If we were to simply list the phases of an Iterative model [4] we would have

a. Requirement Analysis Phase,

b. Design Phase,

c. Implementation & Testing phase and

d. Review Phase.

The process starts with the analysis of the requirements that leads to design and follows with implementation and testing. However, after each cycle a review of the software is done to check if it meets the requirements and if it can be released. If the review phase is a success the software is delivered, else it goes back into the cycle.

3.2 Spiral model

Here in-stage-prototyping and designing elements are combined in order to achieve advantages of bottoms-up and top-down approaches. The methodology has features of waterfall & prototyping models combined. This method is best suited and preferred for complicated, expensive and large scale projects. It utilized the primary structure of the waterfall approach like it's order and phases but the addition of simulation and prototypes is done to the planning phases along with risk assessment phases (Figure 4).

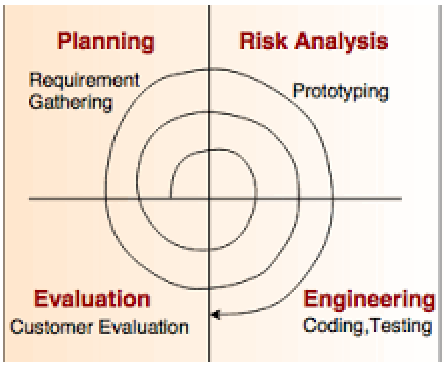

Fig 4: Spiral Model

It utilized in projects with large systems & applications that have built-in smaller segment or phases.

Pros	Cons
• Better estimation (of schedules & budgets etc.) is offered by this approach as it is able to identify critical issues at early stages. • Developers are involved at early stages. • System development is in phases post risk management.	• High time & cost for reaching final products. • Special skills are required for evaluating the assumptions & risks. • Product are quite customized that limit re-usability of products.

3.3 Progressive Development

The model of progressive development is also called incremental delivery or phased implementation. It delivers the software with limited functionality that is termed as the interim version (a common practice in software development), that results in reduced time to market [4]. Models like the PV, traditional V or waterfall may be used for development in each of the phases inside the life cycle of progressive model [4]. Following are the benefits of using the Progressive Development model [1]:

- The software can be used and accessed by the customers right at the first delivery
- Testing becomes more comprehensive as the features & functionality with the highest priority functionality is delivered first.

Yet, it is a difficult task of defining the interim version of the package, specifically since there are no detailed user guidelines and requirement for this stage.

3.4 Sequential Model

In the Sequential model, there is smooth progress of developmental efforts through phases that are very well demarcated [4] e.g. PV Model, Traditional V and the waterfall models.

3.4.1 PV model

In the PV model is the testing is done in the reverse order after the completion of coding, with testing being associated with each developmental activity. Each testing activity is further divided into two sub-activities test Plan & Test execution. The Test plan or the specification is developed for every developmental activity s) are developed along with each development activity as depicted in Figure 5 [1]. In comparison to the traditional V model the testing in PV model can be done at a much earlier stage, resulting reduction of developmental & testing periods which is one of its advantage.

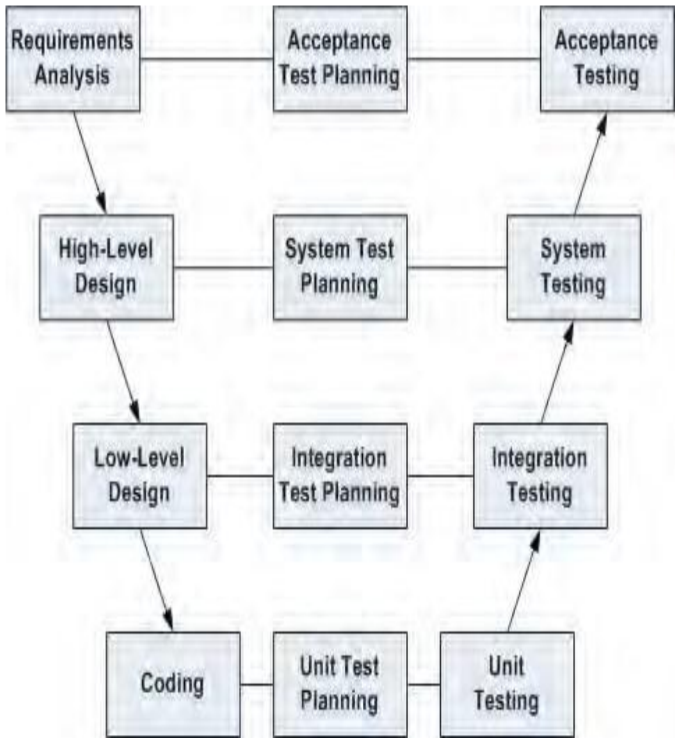

Fig 5. The PV Model

3.4.2 Traditional V-Shape model

It can be called waterfall model extended. However, rather than linear downward movement, post coding & implementation phases the process steps get directed upwards, the process map gives the form of letter V(shown below) hence the name. Primary difference between Waterfall and V-model is that test planning is done earlier in comparison to waterfall model (Figure 6).

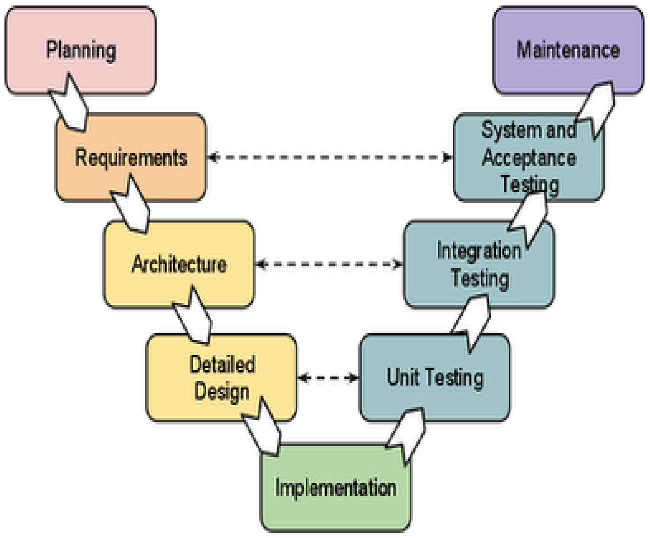

Fig 6: Traditional V-Model

The usage may be summarized as:

- Clearly known & defined Software requirement
- Well known tools and technologies for Software development.

Pros	Cons
• Easy & Simple in using Specific deliverables in each phase • Greater success chance in comparison to waterfall model because of early test plan development. • Works well for where requirements are easily understood. • Product validation & verification in early developmental stages.	• Similar to waterfall model quite inflexible. Scope adjustment is costly & difficult. No prototypes available early since software development is in implementation phase only. No clear path to tackle issues identifies in testing phase. • Time consuming & Costly along with requiring a detailed plan.

3.4.3 Waterfall model

This model like the PV and the traditional V models have a sequential phases that have a downward flow, hence the name “Waterfall”. Even though specific projects may have different phases, but, in general they include “requirement analysis, program designing, coding, testing and Operations” [6]. The process of development is very well documented as it is a requirement to have the documents signed off for each phase before proceeding for the next phase [7]. However, the system proves to be inefficient and ineffective when dealing with changing requirements of the customers. Thus have the same advantages and disadvantages like the PV model with the addition that if there is an issue found then the last phases must be worked upon. This causes extended timelines and budgets [8].

3.5 Agile Methods

This method is very similar to the iterative model that focuses on incremental specification. Designing and implementing [1], which additionally requires complete integration to testing & development [9]. As per the “Manifesto for Agile Software Development”, the method values:

- Interactions and individuals over tools and processes,
- Building functional software over preparing extensive documentations,
- Collaborating with customers over negotiating contracts, and
- Being change ready over just following plans.

The idea is producing high quality and cost effective software in time, while accommodating the changing requirements of the users as well. XP or extreme Programming is the most dominant of the various agile methods of software development. While some of the others are feature driven developments, dynamic system method, adaptive development, scrum, lean development and crystal development methods [10]. Right after a brief period of planning, XP quickly moves to analysis, designing and implementing as depicted in Figure 7 [11]. Using a 1 to 4 week time box, it is ensured that delivery of a enhanced and new software is done post every iteration. Practices and principles of XP include On-site customer presence, sustainable pace, continuous integration, collective ownership, pair programming, refactoring, test-first developments, simple design, small releases and incremental planning [12, 1].

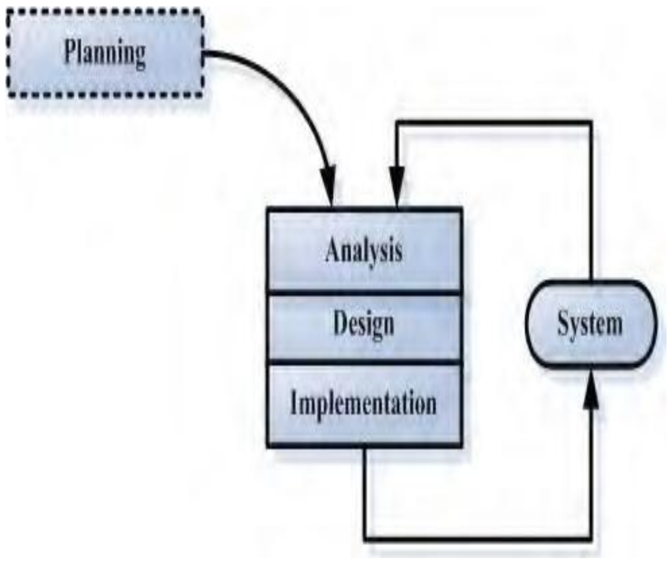

Fig 7: Agile Method - XP-Based

TDD (Test-driven development), or the concept of “writing test cases that then dictate or drive the further development of a class or piece of code” [13] is an essential part of the XP core practices. E.g., Parasoft Corporation requires writing of a minimum 1 test prior to each task of coding [14]. With the ability to change the implementation of a class and to test it again with minimum time & effort, TDD proves to be a powerful tool in dealing with changing requirements of the customer in the project [13]. It was noted by Beck & Andres [12] that respect, courage, feedback, simplicity and communication is encouraged by XP. It was proved by Talby et al. [9], with the help of qualitative and quantitative data, productivity and development quality is improved by agile software developments. In order to maximize success, it is recommended that, while using agile methods, organization must:

(i) Manage Defects,
(ii) Plan activities on Quality
(iii) Utilize professional testers and
(iv) Pay a great deal of attention to designing of tests executing the activities.

4. LATEST TRENDS

After reviewing the most common models and methods of software development we can clearly see 3 emerging trend in field. First, we find that the Agile methods are steadily gaining upon the models based on Life cycle. We see a number of benefits offered by the Agile methods when compared to the life cycle model. We can obtain functional software a lot faster through Agile methods. They are more adaptable to the changes in the user requirements and encourage far better coordination between the stakeholders of the process namely, Project managers, end users, testers, developers and business analysts. It has been described as: “Agile is a systemic change. It drives cost down, quality up and service levels higher by making the entire process leaner, the entire staff more responsible, and the customer more involved” [15]. Additionally, we see a number of empirical studies indicating:

a. The commonly used models based on Life Cycle prove to be ineffective and inefficient for developing complex and large systems [16], and

b. The recent techniques of development like prototyping and Agile methods improve productivity and development quality, significantly.

In fact, a number of established software development companies that have been depending on the Life cycle methods, are nudging towards the Agile methodologies by experimenting with these methodologies [17, 18].

Secondly, we find software testing to have become an integral part of the process of software development. Software testing has become is essential to ensure software quality [19]. Conventionally, we view testing to be an independent and separate phase that comes in the end of the process of software development. Conversely, testing has been established as an activity that goes in sync with the development process instead of being the last activity of the process. This change has been brought about with the evolution of software development technology. Testing has been argued in [7] as "an integral activity in software development" & it is suggested that "testing should be included early in software development". In [20] it is also suggested "testing should be performed throughout the software life cycle".

Thirdly, we discovered that because of the increasing importance of testing in the software development, testers have started to play a crucial role in the process as well. We see roles of testers expanding in 2 ways to include:

1. Quality Assurance [21] and code validation & verification [22]
2. Now engagement of testers is throughout and much earlier in the process of software development, and this adds many benefits to the performance of the project team.

For example, in [23] quantitative evidence is presented to establish that without compromising on the software quality; the cycle time & cost effectiveness and project performance is improved by initiating testing at a much earlier time in the process of software development. It also enables catching of defects earlier on, which reduces development cost. As it is much more expensive to fix defects found at the later stages of the software development [24].

5. EFFECTS

The latest trends of software development techniques prevalent have major impact and effects especially on the International Software research community. Firstly, with the adoption of agile methods by major organizations more empirical studies are required for clarification of the effects on these methodologies [25]. We may be research on the following questions:

1. What is agile method's impact on Product Quality?
2. What is agile method's impact on Job satisfaction of the team members?
3. What is agile method's impact on working relationship of team members?
4. What is agile method's impact on schedule and project budget adherence?
5. What is agile method's impact on project success levels?

If the answers and these questions are better understood then development companies would be able to make better decisions on implementing the agile methods of software development.

Secondly, with software testing assuming a critical role in the software development we would require empirical studies for answering the following questions from research:

1. Which is the best possible approach to integrate testing into the process of software development?
2. What should be the intensity, frequency and magnitude of testing employed in a development project?
3. How would project success be affected by adopting the above-mentioned approaches of testing?

Thirdly, owing to the increased vitality of software testers in development process, we need empirical studies to answer the following:

1. How is the working life of the testers impacted by the growing importance of their role in the process of software development?
2. What is the impact of this change on the working relations of developers and testers?
3. Which additional skillset is required for preparing the testers for their expanded and new role?

6. CONCLUSION

The paper reviewed methodologies of software development, highlighted the emerging trends and discussed the effects and impact of these trends on software development. The paper contributes in three major ways:

a. Discussion about the methodologies of software development would help students and educators in gaining a deeper understanding of these methodologies.

b. Highlighting of the emerging trends in the field would help guide development practitioners in making better career, tactical development and strategic decisions.

c. The discussion on impact and effects can be helpful for future researchers in field software development.

We do hope the paper would help in generating relevant actions, inspiring creativity and instilling knowledge for the betterment of the software development field.

7. REFERENCES

[1] Sommerville, I. (2011) Software engineering, 9th edition. Boston, MA: Addison-Wesley ISBN 10: 0-13-703515-2

[2] W. W. Royce, Managing the development of large software systems: concepts and techniques, Proceedings of the 9th international conference on Software Engineering, p.328-338, March 1987, Monterey, California, USA.

[3] Navid Hashemi Taba (2012) Improving Software Quality Using a Defect Management-Oriented (DEMAO) Software Inspection Model, 2012 Sixth Asia Modelling Symposium, DOI 10.1109/AMS.2012.51

[4] Xihui Zhang, Tao Hu, Hua Dai and Xiang Li, 'Software Development Methodologies, Trends and Implications: A Testing Centric View, Information Technology

Journal, Volume 9 (8): 1747-1753, 2010.

[5] Ahmed Mateen, Muhammad Azeem Akbar, Mohammad Shafiq, 'AZ Model for Software Development', International Journal of Computer Applications (0975 – 8887) Volume 151 – No.6, October 2016

[6] Munassar, N. M. A. and Govardhan, A. 2010. A Comparison between Five Models of Software Engineering. IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 5, pp: 94-101, September 2010.

[7] Bassil Y. 2012. A simulation model for the waterfall software development life cycle. International Journal of Engineering & Technology, 2(5): 1-7.

[8] Majumdar A., Masiwal, G. and Chawan, P. M. 2012. Analysis of Various Software Process Models, International Journal of Engineering Research and Applications, Vol. 2, No. 3, 2012, pp: 2015-2021

[9] Hurst, J. 2014. Comparing Software Development Life Cycles, SANNS Software Security.

[10] Taya, S. and Gupta, S. 2011. Comparative Analysis of Software Development Life Cycle Models, IJCST Vol. 2, Issue 4, Oct.-Dec. 2011.

[11] Shmuel Ur, Elad Yom-Tov and Paul Wernick, An Open Source Simulation Model of Software Development and Testing, Hardware and Software, Verification and Testing, Lecture Notes in Computer Science, Springer, vol. 4383, pp. 124-137, 2007

[12] B. Boehm and K.J. Sullivan, "Software Economics: Status and Prospects," Special Millenium Issue, Information and Software Technology, 2000.

[13] Leung, H., and Fan, Z., Software Cost Estimation. Handbook of Software Engineering, Hong Kong Polytechnic University 2002

[14] Alan M. Davis, Edward H. Bersoff and Edward R. Comer, A Strategy for comparing Alternative Software Development Life Cycle Models, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 14, NO. 10, OCTOBER 1988

[15] Y. Chernak, "Validating and improving test-case effectiveness," Software, IEEE , vol.18, no.1, pp.81-86, Jan/Feb 2001doi: 10.1109/52.903172

[16] M. V. Mantyla, C. Lassenius, " What Types of Defects Are Really Discovered in Code Reviews?," Software Engineering, IEEE Transactions on , vol.35, no.3, pp.430-448, May-June 2009 doi: 10.1109/TSE.2008.71

[17] D. E. Perry, A. Porter, M. W. Wade, L. G. Votta, and J. Perpich, " Reducing inspection interval in large-scale software development," IEEE Transactions on Software Engineering, 28(7), 695-695-705. doi:10.1109/TSE.2002. 1019483

[18] O. Laitenberger, and J. DeBaud,"An Encompassing Life Cycle Centric Survey of Software Inspection," Journal of Systems and Software, Vol. 50, 2000, pp. 5-31.

[19] A. Porter, H. P. Siy, C. A. Toman, and L. G., "An experiment to assess the cost-benefits of code inspections in large scale software development,"IEEE Transactions on Software Engineering, 23(6), 1997, 329-329-346. doi:10.1109

[20] M. Agrawal, K. Chari, "Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects," Software Engineering, IEEE Transactions on , vol.33, no.3, pp.145-156, March 2007 doi: 10.1109/TSE.2007.29

[21] D. L Parnas, and M. Lawford, "Inspection's role in software quality assurance," IEEE Software, 20(4), 16-16-20. doi:10.1109/MS.2003.1207449

[22] C. K. Tyran, "A Software Inspection Exercise for the Systems Analysis and Design Course," Journal of Information Systems Education, 17(3), 2006, Pp. 341-351.

[23] T. Dyb, B.A. Kitchenham, M. Jrgensen, "Evidence-Based Software Engineering for Practitioners", IEEE Software, vol. 22, no. 1, pp. 58-65, 2005.

[24] T. Berling, T. Thelin, "An Industrial Case Study of the Verification and Validation Activities", Proc. 9th Int'l Software Metrics Symp., pp. 226-238, 2003.

[25] A. Aurum, H. Petersson, C. Wohlin, "State-of-the-Art: Software Inspections after 25 Years", Software Testing Verification and Reliability, vol. 12, no. 3, pp. 133-154, 2002.