Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 203 (2022) 775–782

Procedia Computer Science

www.elsevier.com/locate/procedia

The second International Workshop on Edge IA-IoT for Smart Agriculture
August 9-11, 2022, Niagara Falls, Canada

Software Testing from an Agile and Traditional view

Soukaina Najihi[a,]*, Sakina Elhadi[a], Rachida Ait Abdelouahid[a], Abdelaziz Marzak[a]

[a]*Laboratory of Information Technology and Modeling, Faculty of Sciences Ben M'sik, Hassan II University- Casablanca, BP 7955 Sidi Othman Casablanca, Morocco*

Abstract

Project management has long been a challenge for Information Technology (IT) organizations to learn how to manage projects effectively and efficiently while balancing the iron triangle: cost, time and scope without sacrificing quality. Agile project management (APM) and traditional project management (TPM) are two different approaches related to software development, the core tenet of TPM is that projects are relatively simple, predictable, and linear with well-defined boundaries, whereas APM has emerged as a new methodology for managing high-risk, time-sensitive and scope flexible projects. While it is undeniable that the use of agile project management is increasing, the traditional opposite side continues to exist. The agile approach has an impact on all phases of software development, and they must adapt to this new way of thinking. Software testing is also impacted as a fundamental link in the software development life cycle that ensures quality. Software testing is a vital component of software Quality Assurance (QA) since it serves as the final review of the specification, design, and code. Has software testing evolved along with the evolution of project management? Is there a difference between a traditional and an agile software testing? This paper answers these questions by comparing software testing in the two approaches.

Keywords: Software Development Life Cycle (SDLC); Software Testing; Agile Software Development; Traditional Software Development; DevOps; Continuous Integration; Continuous Development; Continuous Testing.

* Corresponding author. Tel.: +33 6 64 73 36 14.
E-mail address: soukaina.najihi@gmail.com

1877-0509
10.1016/j.procs.2022.07.116

1. Introduction

In the software development process, different approaches are used. "There is no single development, in either technology or in management technique" [1]. A software development process is also called a Software Development Life Cycle (SDLC) [2]. IT Organizations building software solutions confront the challenging task of selecting the most appropriate approach. Several of them have opted for Agile, while others have opted for the traditional way. SDLC selection and adoption is critical because it maximizes the organization's likelihood of effectively delivering software; therefore, selecting and adopting the proper SDLC is a management choice with long term consequences and percussions [3]. While software development processes differ, the goal of IT organizations remains the same: to provide a higher-quality service at the lowest cost while meeting deadlines.

A software program or information system is created to carry out a specific set of tasks. Typically, this assignment that the system will do produces well-defined outcomes that involve complex computing and processing. To ensure that the final product has a high degree of integrity and robustness, as well as user approval, it is consequently a difficult and laborious task to oversee the entire development process. In order to attain the aforementioned 'characteristics of a successful system,' a systematic development process that emphasizes the awareness of the scope and complexity of the entire development process is required.

Project management (PM) approaches must keep up with the need for the creation of increasingly sophisticated software products and the pressure to market them quickly. Through the Agile methodology, the classical methodologies have been changed to a higher level.

Agile methodologies rose as a reaction to the difficulties of applying traditional plan-driven methodologies [4] [5]. While Traditional Software Development (TSD) methods claim their support for detailed planning, large-scale design, and thorough documentation and do not scale to the challenges brought by digital transformation, Agile Software Development (ASD) assumes short periods of time between each delivery to ensure early customer participation and continuous software delivery, which makes it react quickly to changing customer needs and, at the same time, gives the organization a possibility to preserve its stability and take time to carry out large digital transformation changes.

ASD approaches enhance system productivity and quality. In terms of quality, the DevOps (Development and Operations) movement has emerged that aims to take this line of reasoning to an even higher level. This movement aims to break the traditional culture where teams work in silos without any interaction between them [6], and the goal is to create a culture of collaboration between development and operations teams that allows an increase in the flow of completed work [7]. It defines new processes and standards requirements where Continuous Integration (CI) principles have filled the gap while improving the quality of the system continuously by frequently merging code [8]. Continuous Delivery (CD) has accelerated software delivery by automatically releasing code changes to a testing and/or production environment after the build phase [9].

Agile and DevOps collaborate to provide clients with high-quality value. Modern Software Development (MSD-agile and DevOps) is based on the guarantee that all developed software components will be of superior quality. MSD methods make QA the heart of the software development process.

Continuously producing high-quality software is the greatest challenge for software development organizations. Software testing has its own significance in the software development life cycle, and it is one of the most essential methods for confirming the quality of a software product. A proper testing strategy is essential for a successful project. It can save substantial time and substantially improve the quality of software products. Extensive testing decreases post-delivery faults and maintenance expenses, which increases customer satisfaction. Testing is running the software under specific controlled conditions, which means first verifying that it behaves "as specified", second detecting errors, and third confirming that what has been specified is what the user actually needs [10]. Testing gives accurate information about how a system works, which helps make high-quality software.

Software testing has effectively grown over time and has offered effective and ongoing support for software quality enhancements. It remains a struggle for upper management to persuade clients of the software project development lifecycle's significant resource consumption. Furthermore, software testing is a particularly costly component of the software development process, accounting for an estimated 40% of total development costs [11].

Testing in Agile projects differs from traditional testing because of the iterative nature of this method. Even though Agile testing and traditional testing are different in many ways, their goals are the same: to make sure that the requirements are met.

The literature on agile and traditional software development contains a large number of research works, but few of them have performed comparative studies of testing in Modern and Traditional software development approaches, and none appear to have provided a complete study by including DevOps as a comparison criterion.

The main goal of this paper is to shed light on the evolution of software testing, starting from Traditional techniques towards Agile and continuous delivery via DevOps, as well as the challenges [3] that come with them.

The remainder of the paper is structured as follows. Section 2 defines "Traditional and Agile software development" and the difference between the two. Section 3 presents a comparison between testing in Traditional and Agile methodologies. Next, in Section 4, we present our discussion. We conclude our study in Section 5.

2. Agile Software Development Vs. Traditional Software Development

Software has been around for about 50 years. Software development began as a messy process known as "code and fix". The software was created without much planning, and the system design was influenced by several quick decisions. This worked for small systems, but as systems evolved, adding new features and fixing issues got more challenging. This development style was employed for many years until a methodology was developed [12].
Methodologies enforce a disciplined approach to software development to make it more predictable and efficient.
Agile and Traditional (see Fig. 1), such as waterfall, are two fundamental and widely used software development approaches that have been used in software development projects for decades.

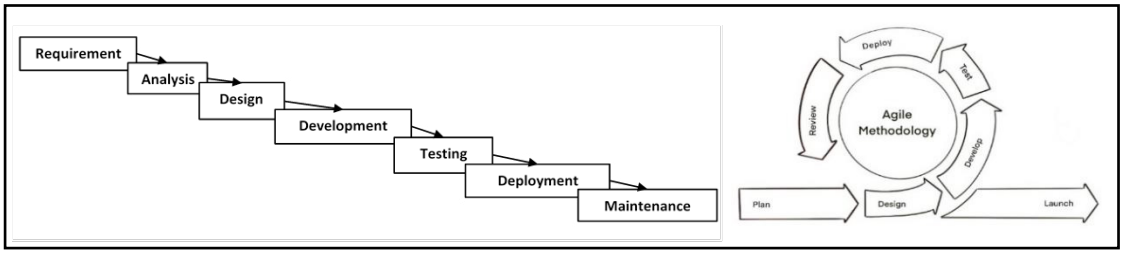

Fig. 1. Software Development Lifecycle : Agile Vs. Traditional.

2.1. Traditional Software Development

Traditional software development approaches like Waterfall, V-Model, and Rational Unified Process (RUP) are classified as heavyweight methodologies [4]. The TSD processes start with collecting and documenting all requirements, then go on to architectural and high-level design development and assessment. Heavyweight methodologies got its name from its weighty characteristics [13] [14].

In traditional methodology, all the following phases: requirements, analysis, design, development, testing, deployment, and maintenance, are carried out simultaneously, and testing is generally carried out at the end of the process [13] (see Fig. 1). This means that once each of the eight steps has been accomplished, developers progress to the next stage, and they cannot return to a previous step, at least not without scrapping the entire project and starting over. For a project to succeed and because of the extremely low rate of change and error, a project outcome and comprehensive strategy must be formed in the beginning and then followed carefully.

In traditional projects, integration occurs between development and testing. When the system is put together in order to be functionally tested as a whole, this is often the time of frustration, as things may not fit or work together as planned, and the team may spend days or even weeks making the appropriate changes [15]. Depending on the delivery schedule, the project may encounter difficulties at this stage, as the objective was to integrate the components into a fully functional system within a specified time range. However, there is now a delay due to issues regarding integration. Therefore, there are valid reasons to integrate as frequently as possible. This is what we call continuous integration. CI means a team often integrates changes. This allows all developers to benefit from a

change quickly and permits early testing in their real context. Continuous integration means each member should integrate team modifications to find incompatible changes early. Frequent integration reduces overall integration costs because incompatible changes are recognized and fixed early [16].

In Traditional Software Development, integration is frequently performed by a distinct team, due to organizational silos. The lack of communication between the teams might result in the delayed detection of problems and, thus, the postponement of project deadlines. Therefore, a new idea, DevOps, has arisen that breaks through the walls between development and operations, something that traditional plan-driven techniques lacked.

The principal benefits of CI methods are risk reduction and bug-free, reliable software, which lowers the barrier to frequent delivery. Throughout development, the team ensures that its software is always ready for release, during which the software is transferred to a testing and/or production environment. Unlike TSD, where the deployment phase is realized only once in a SDLC, the key aim of Continuous Delivery and DevOps is to provide more reliable apps more quickly and frequently to meet client and business requirements securely and sustainably. There is a significant trend in CD investment due to its benefits, including improved production and efficiency, reliable releases, improved customer satisfaction, expedited time to market, and developing the correct product.

DevOps and continuous delivery are yet another agile subset. Due to Traditional Software Development rigidity and linearity, DevOps practices are incompatible with TSD since one of the main aims of DevOps is to provide new software features frequently and promptly.

2.2. Agile Software Development

Agile is a word that refers to software development methodologies that emphasize incremental delivery (see Fig. 1), team communication, continuous planning, and continuous learning, rather than attempting to release everything all at once near the end [17].

Agile emphasizes lean processes and the creation of minimum viable products (MVP) [18] that go through multiple iterations before being finalized. It is done in a highly collaborative manner to produce high-quality software that meets the changing needs of its stakeholders. Continuous feedback is gathered and applied, and the overall process is much more dynamic, with everyone working toward a common goal. Here are the characteristics of the agile method [19]:

- Iterative and continuous planning
- Iterative and Incremental Product delivery
- Customer able to view constantly the subset of the product being developed before launch.
- Foster business and customer collaboration throughout the development cycle
- Risk reduction through efficient and timely change management
- Transparent status reporting through frequent product demos
- Process refinement through retrospection discussions

Agile assisted the software industry in responding to client requests and keeping up with change, whereas DevOps is the key to delivering value to end users. DevOps involves the collaboration of the entire organization to deliver customer value. DevOps enables you to streamline the software delivery pipeline, which entails delivering software from a concept to the end user.

By implementing DevOps practices, enterprises such as Netflix, Amazon, and Google are hitting performance levels that were inconceivable with Agile. Beginning as new enterprises in the late 1990s, by the late 2000s they, along with other online expert corporations, were constructing a large-scale, complicated framework to assist their expanding clientele. These organizations do not deploy annually or even weekly. Instead, they deploy many times each day while providing a consistent and dependable customer experience.

From one perspective, the objective of operations is to assure nonstop client service by providing a stable and dependable workplace. Alternatively, the purpose of development is to continue developing and releasing new enhancements in production or in current services using an agile methodology.

The magical combination of Agile and DevOps leads to an increased velocity of change compared to traditional development.

2.3. Difference between agile and traditional approaches

Agile methodology is much more than a "new process" in software development. It is a culture change among IT organizations, especially within development teams. The table below summarizes the differences between Agile and Traditional approaches [14]:

Table 1. Difference between Agile and Traditional Methodologies.

Characteristic	Agile approach	Traditional approach
Requirements	input is required throughout the development process	Clearly defined and thoroughly reviewed from the initial phase of Software Development Life Cycle (SDLC). The rate of change is low.
Customer participation	The customer is highly involved	Lack of interaction with the customer; the client only sees the product at the end of the project.
Documentation	Software increment is placed above exhaustive documentation. Agile documentation is scalable, and team based.	Required formal documentation
Project size	Small to medium	large
Team members	Between 5-11	More than 15
Team structure	Self-managed, multi-skilled team	Team members have specific roles and are not interchangeable.
Project organization	Iterative	Linear
Project characteristics	Flexible, adaptative	Rigid, predictable
Focus	value	process
Teams' collaboration	Flexible organization	Organization silos with defined boundaries
DevOps	Compatible	Incompatible
Delivery of product	Planned	Frequent—loose plan
QA function	Formalized—Separated	Informal—Embedded
Code integration	Continuous integration	only when the development phase is complete
Releases/month	1 every 3 weeks (with DevOps : several times a day)	once in the SDLC

Although software development time and costs are reduced [13], but the agile development methodology is not for everyone. IT organizations must be ready to take software delivery iteratively. They need discipline and an adoptive culture. Up-front work to define, plan, and architect the business solution and train the teams is a must for the agile development process to be successful.

The testing team is one of the major teams that requires complete culture change and is responsible for the most important function in the agile process. Sprint testing is part of the sprint schedule, there is no additional time for testing. It is very important that the testing team participate in sprint planning, thus time is set aside to test developed code [20].

3. How is Agile Testing different from Traditional Testing?

Agile development requires team members to work collaboratively, be adaptable to changes, and be able to change course when needed. This is particularly important for team members who can work with design and development teams to give early feedback and test as the sprints progress. Emphasis is on coordination and shared responsibility to develop working software. Listed below are some differences between testing from Traditional and Agile view [20] [21] [22].

Table 2. A comparison between traditional testing and agile testing.

Factor	Agile	Traditional
Team	Testers are integrated into the development team.	Separate test teams and testers are very skilled at testing.
Process	Testing preparation begins with user stories and testing continues with development. Focus is to test high value areas first.	Focus is on test preparation, execution, documentation, and communication through documentation. Typically, test teams are heavily involved after development is complete
Communication	Continuous stakeholder communication.	Infrequent communication with client, development teams and stakeholders.
Accommodation of Changes	Adaptable to changes. Changes are expected as part of the process and cause minimal delays.	Works on pre-defined scope, and changes to scope usually cause delays in the whole schedule.
Planning	Rough planning is enough.	Needs very formal planning.
Cost	Typically, less costs and cost of change is minimal	More costs and high cost of change
Testing phases	Depends on size and type of the projects.	Multiple test phases
Development	Focus is on end-to-end solution deployment including testing.	Business solutions are divided into multiple pieces – presentation, persistency, cots etc.
Tester skills	Testers have knowledge of software testing and software development and are responsive to changes.	Experienced and skilled in testing, test processes.
Automated testing	Fundamental	Optional
Testing	Every iteration (DevOps : Continuous testing)	After coding phase completed
Characteristic	Agile approach	Traditional approach
Requirements	input is required throughout the development process	Clearly defined and thoroughly reviewed from the initial phase of Software Development Life Cycle (SDLC). The rate of change is low.

In agile, testing is not a separate phase and testers are involved in the process right from the requirements gathering phase. Testing is a continuous process engraved within the incremental cycle. Since development happens in iterations, usually in 2-3 week cycles, the testing team works continuously in cycles, in tandem with the developers and Product Manager (aka Product Owner, who is a member of a product development team who manages the product backlog to reach the desired output). These iterations are highly adaptive, with requirements proofing, development, integration, and testing taking place throughout. Testing happens throughout this iteration, and not just as a final activity.

In agile, testing is the responsibility of the entire team (which consists of business stakeholders, PM, developers, and testers), unlike in traditional approaches such as waterfall where it is only the QA team that does test. Developers' unit test the user stories, testers test the user stories for functional correctness, and business stakeholders and PO's test the features from the user's standpoint. This way, testing is critical in providing faster feedback in the shorter execution window due to the idea of very small iterations and each iteration producing potential shippable software [18].

In some Agile practices like Extreme Programming, pairing is used, which involves developers and testers working together to develop and test features within the iteration.

In traditional plan-driven approaches, the focus of testing is on finding and resolving defects. However, in Agile, the focus of testing is on providing value to the users by detecting defects as well as preventing them through continuous collaboration with everyone in the Agile team, and by being proactive in the ceremonies.

Also, in traditional approaches, final acceptance testing is typically done at the end of the project. In agile, acceptance testing is done incrementally at the end of each sprint on the items produced in the sprint.

4. Discussion

The traditional method tends to see verification and quality as independent processes, to be considered only at the end of software development, just prior to release. Test activities are performed late in the development phase to detect but not avoid failures. After development, the development team hands over both high-level and low-level technical documentation to the testing team. Detailed requirements and a test strategy are developed and maintained throughout the life cycle. The testing process is driven by the requirements perspective, and test case automation is optional.

When the testing phase begins, testers test the product and report defects prior to deployment. The development team corrects any faults with the optimal solution. This phase is effectively the merging of all types of testing (unit, functional, and integration) into a single phase that occurs following the completion of the majority of significant system development.

Traditional testing is based on the idea that the Traditional process is repeated and can be predicted. It standardizes procedures by assigning distinct jobs to individuals based on skills. The development team creates the product, while the QA team inspects it; Only the QA team is responsible for quality. However, despite the traditional model's seeming clarity, it lacks adaptability. The technique is time-consuming because the team executes tasks in a predetermined order.

Agile Methodologies, on the other hand, aim to correct the rigidity of Traditional methods. It is a team-based strategy. Agile testing relates to the execution of testing activity from the early stages of agile projects. Agile testing follows the principles of the agile software development methodology and ensures it is an integral part of development. It consists of a cross-functional team where testers along with the developers together ensure the quality and business functionalities are delivered in a seamless way. Agile software development follows an iterative approach, and requires frequent testing on a regular basis, may be daily. Manual testing on a daily or weekly basis throughout the development process would be inefficient. Automated testing can run through a large series of test cases rapidly and efficiently.

Due to the rapid changes and the limited time frame, testers frequently struggle. They are constantly attempting to catch up with the developers. As difficult as it is for testers to maintain quality, the operations team is in an even worse position. These teams typically reside in another department.

They are responsible for maintaining the datacenter, deploying the product generated by the development team, and ensuring that everything continues to function properly. This results in silos, which are comprised of different teams in separate divisions reporting to separate supervisors with distinct responsibilities. The operations team is accountable for guaranteeing the program's stability. The development team is responsible for introducing new features as rapidly as possible. The testers are trapped in the middle, while the business is aware that it takes too long for the necessary resources to become available. This is where DevOps comes in to enhance the process and improve the quality, by introducing new practices : continuous integration, continuous testing, and continuous delivery.

There are many advantages to CI, CT and CD. First, errors may be corrected swiftly while the context is still fresh in the minds of the developers and before these errors cause further complications. Additionally, the fundamental reasons that lead to the difficulties can be recognized and eliminated. In addition, the testing process is typically somewhat automated and test cases are prioritized. Continuous testing recommends that every test case be automated. Today's technology can automate any repetitive task. In most situations, the manual testing procedure can be automated. Software delivery process should be able to perform the test suite on every software build created automatically and without human interaction, thus progressing towards the ultimate objective of being able to send a quality release rapidly. This whole notion of continuous testing not only brings the testing process to the beginning of the development cycle but also allows testing on a production-like system.

5. Conclusion

Agile methods were developed as a response to the issues that traditional methodologies had with defining requirements and delivering a product that turned out to be not what the end user actually wanted and needed.
Agile methodology is seeing increasing widespread adoption for projects with frequently changing requirements. In

Agile development, testing is integrated throughout the lifecycle, or the software is tested throughout its development. Agile testing can be taken for testing for those who want to find defects early in the SDLC and who want to deliver a quality product. Agile projects are in fact an excellent opportunity for QA to take leadership of the agile processes, which bridges the gap between users and developers, understanding both what is required, how it can be achieved, and how it can be assured prior to deployment.
Agile testing has solved many problems of traditional methodologies such as waterfall, but it cannot be denied that there are many challenges facing the testing team in an agile environment that make the task of testing more complicated. These challenges and how to overcome them will be the subject of an upcoming study.

References

[1] Brooks, "No Silver Bullet Essence and Accidents of Software Engineering," *Computer*, vol. 20, no. 4, pp. 10–19, Apr. 1987, doi: 10.1109/MC.1987.1663532.
[2] C. Lane, "Systems Development Lifecycle: Objectives and Requirements is," p. 60.
[3] R. K. Gupta, P. Manikreddy, and A. GV, "Challenges in Adapting Agile Testing in a Legacy Product," doi: 10.1109/ICGSE.2016.21.
[4] S. Balaji, "Waterfall vs V-Model vs Agile: A comparative study on SDLC," . *Vol.*, no. 1, p. 5, 2012.
[5] Y. B. Leau, W. K. Loo, W. Y. Tham, and S. F. Tan, "Software Development Life Cycle Agile vs Traditional Approaches," p. 6.
[6] W. P. Luz, G. Pinto, and R. Bonifácio, "Building a Collaborative Culture: A Grounded Theory of Well Succeeded DevOps Adoption in Practice," doi: 10.1145/3239235.3240299.
[7] A. Hemon, B. Lyonnet, F. Rowe, and B. Fitzgerald, "From Agile to DevOps: Smart Skills and Collaborations," *Inf Syst Front*, vol. 22, no. 4, pp. 927–945, Aug. 2020, doi: 10.1007/s10796-019-09905-1.
[8] S. Stolberg, "Enabling Agile Testing through Continuous Integration," in *2009 Agile Conference*, Chicago, USA, Aug. 2009, pp. 369–374. doi: 10.1109/AGILE.2009.16.
[9] S. A. I. B. S. Arachchi and I. Perera, "Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management,", May 2018, pp. 156–161. doi: 10.1109/MERCon.2018.8421965.
[10] S. K. Singh and A. Singh, *Software Testing*. Vandana Publications.
[11] K. Pal, "Framework for Reusable Test Case Generation in Software Systems Testing," *Software Engineering for Agile Application Development*, 2020.
[12] T. Mens, "Introduction and Roadmap: History and Challenges of Software Evolution," in *Software Evolution*, T. Mens and S. Demeyer, Eds. Berlin, Heidelberg: Springer, 2008, pp. 1–11. doi: 10.1007/978-3-540-76440-3_1.
[13] A. K. M. Z. Islam and Dr. A. Ferworn, "A Comparison between Agile and Traditional Software Development Methodologies," *GJCST*, pp. 7–42, Dec. 2020, doi: 10.34257/GJCSTCVOL20IS2PG7.
[14] W. Van Casteren, *The Waterfall Model and the Agile Methodologies : A comparison by project characteristics*. 2017. doi: 10.13140/RG.2.2.36825.72805.
[15] L. Bendix and T. Ekman, "Software Configuration Management in Agile Development," p. 19.
[16] M. Virmani, "Understanding DevOps & bridging the gap from continuous integration to continuous delivery,", May 2015, pp. 78–82. doi: 10.1109/INTECH.2015.7173368.
[17] D. Cohen, M. Lindvall, and P. Costa, "DACS State-ofthe-Art / Practice Report Agile Software Development,"
[18] G. Schuh, C. Doelle, and S. Schloesser, "Agile Prototyping for technical systems – Towards an adaption of the Minimum Viable Product principle," presented at the NordDesign 2018.
[19] K. Beck *et al.*, "Manifesto for Agile Software Development," p. 10.
[20] J. R. Penmetsa, "Agile Testing," in *Trends in Software Testing*, H. Mohanty, J. R. Mohanty, and A. Balakrishnan, Eds. Singapore: Springer, 2017, pp. 19–33. doi: 10.1007/978-981-10-1415-4_2.
[21] J. Watkins, *Agile Testing: How to Succeed in an Extreme Testing Environment*. Cambridge University Press, 2009.
[22] C. J. Gil Arrieta, J. L. Díaz Martínez, M. Orozco Bohórquez, A. K. De La Hoz Manotas, E. M. De La Hoz Correa, and R. C. Morales Ortega, "Agile testing practices in software quality: State of the art review," Oct. 2016.