Agile Software Development: Methodologies and Trends

Article *in* International Journal of Interactive Mobile Technologies (iJIM) · July 2020

DOI: 10.3991/ijim.v14i11.13269

3 authors:

Samar Al-Saqqa
University of Jordan
20 PUBLICATIONS **591** CITATIONS
SEE PROFILE

Samer Sawalha
Princess Sumaya University for Technology
8 PUBLICATIONS **264** CITATIONS
SEE PROFILE

Hiba Abdel-Nabi
Princess Sumaya University for Technology
23 PUBLICATIONS **493** CITATIONS
SEE PROFILE

Agile Software Development: Methodologies and Trends

https://doi.org/10.3991/ijim.v14i11.13269

Samar Al-Saqqa (✉)
The University of Jordan, Amman, Jordan
s.alsaqqa@ju.edu.jo

Samer Sawalha, Hiba AbdelNabi
Princess Sumaya University for Technology, Amman, Jordan

Abstract—Software engineering is a discipline that undergone many improvements that aims to keep up with the new advancements in technologies and the modern business requirements through developing effective approaches to reach the final software product, agile software development is one of these successful approaches. Agile software development is a lightweight approach that was proposed to overcome the convolutional development methods' limitations and to reduce the overhead and the cost while providing flexibility to adopt the changes in requirements at any stage, this is done by managing the tasks and their coordination through a certain set of values and principles. In this work, a comprehensive review that outlines the main agile values and principles, and states the key differences that distinguish agile methods over the traditional ones are presented. Then a discussion of the most popular agile methodologies; their life cycles, their roles, and their advantages and disadvantages are outlined. The recent state of art trends that adopts agile development especially in cloud computing, big data, and coordination are also explored. And finally, this work highlights how to choose the best suitable agile methodology that must be selected according to the task at hand, how sensitive the product is and the organization structure.

Keywords—Agile Development, Agile Methods, Big Data, Coordination, Cloud Computing.

1 Introduction

The Software Engineering expression is a very old expression; it was founded in 1968 by a NATO study group, they focused on the software problems and they encouraged to use the software in many activities in our lives. Software engineering is a field of the engineering family, because that the software product needs to pass by all of the analysis, design and development processes same as other engineering fields such as the electrical engineering, mechanical engineering, and civil engineering. The software also is developed to be used in a particular purpose same as other engineering fields.

[1] IEEE defined the software engineering as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software [2]. It can be seen from the definitions that the final product is important but the approaches and the steps that are needed to produce the software must be done in a systematic, disciplined and quantifiable way in all of the software life cycles. By following these approaches' characteristics, the resulted software will be more reliable, easy to maintain and meet the software requirements, especially when the size of the software is very huge and containing a lot of functionalities [1]. Software development and engineering is a cooperative work, tasks may be distributed over different teams, these tasks must be managed and ordered based on some criteria. Tasks can be done in parallel, but there are some tasks cannot be started before another task is ended. So, the need of coordination between these tasks, processes, and teams is a must to gain the best software or product with the minimal cost. [3]

There are many traditional software development methods and approaches, such as the waterfall approach, iterative and incremental approach, spiral approach, evolutionary approach, etc. [4]. These approaches sometimes are called planned software development approaches, or heavyweight approaches. These approaches are very useful when there is a need to develop a huge complex software, it can help to eliminate old fashion informal software development and deliver high quality software in a systematic way which meets the user's requirements in a predefined limited time [5].

The problem of the traditional software development methods that they need a very heavy process, such as [1]:

1. The project's plan must be done in advance.
2. The written software's requirements.
3. Fully design that satisfy the written requirements.
4. Build the code of the software that satisfies all the written requirements and the designs.
5. Fully test the software and check if it satisfies the requirements and the design.

Many projects that follow the traditional software development methods addressed major problems, especially in the maintenance and the changes based on the user's requests. Some of these changes may lead to major changes which are considered as a big problem in software development. Because all of that the need for a lightweight software development method is needed, the main goal of these methods is to speed up the development and effectively respond to the requested changes. This lightweight software development method is called Agile software development methods. [1].

In the literature, many reviews are conducted in order to capture the current state of the art of the literature related to Agile software development, but there is no comprehensive review that includes Agile methodologies and new trends with agile such as cloud computing, big data, and coordination. In this review, thirty-two studies were considered, ranging from 2011 to 2018. The rest of this paper is organized as follows. In section 2, we present the Agile development process. In section 3, we present software development versus traditional software development. In section 4, we describe Agile methods. In section 5, we present big data systems using Agile development. In

section 6, we present cloud computing and Agile. In section 7, we describe coordination and agile development. In section 8, we present the conclusion.

2 Agile Development Process

Agile is a wide umbrella of software development beliefs. It is a conceptual framework for software engineering that begins with a starting planning phase and follows the road toward the deployment phase with iterative and incremental interactions throughout the life-cycle of the project. The initial goal for the agile methods is to reduce the overhead in the software development process with the ability to adopt the changes without risking the process or without excessive rework.

A light weight [6] adaptive methods are originated and promoted through an official alliance of 17 software engineering consultants that held in 2001 and resulted in the publication of 'Agile Software Development manifesto' [7] in which a set of values and principles in software and system agility are outlined. In this philosophy, four values and twelve principles supported and constituted the essence to be agile. These values and principles provide the basis to guide the software development process [8] and to give the distinguishable characteristics for any method with the agility feature. The Manifesto states that in any situation where a choice has to be made, a priority is given toward the items on the left of each core of its value rather than the ones in its right. These four values are as follows:

1. **Individuals and interactions over processes and tools:** The first value in the manifesto implies that emphasizing in the abstract formal processes and their technical surrounding environment as a key factors in the software developing is incorrect, the more important is the communication, interaction and the quality of the human software developers that these factors serve [8].
2. **Working software over comprehensive documentation:** The documentation of any agile software development process is a vital and valuable component but the amount of time and resources that are given to it must be controlled and optimized not to overwhelm the software development process. The usability of the intensive documentation is one of the traditional development limitations [9], since writing and up-to-date synchronization of the documentations with the code is a time-consuming process especially if the requirements are changing very frequent. Moreover, these documents are rarely used beyond the initial deployment phase. Therefore, the documentation that matters in agile are the documentation that serves the working software [8] and adds value to the process. The manifesto states that the real progress in the agile methods are measured through the tested working software rather than the documentation since it is less ambiguous and can answer immediately if they meet the demands or not [10].
3. **Customer collaboration over contract negotiation:** The agile software developments are invented to cope with the changes in the requirements at any stage, consequently a customer feedback, negotiation and collaboration with the development team are required on a frequent basis throughout any process to reach the real required needs of the customers rather than through a formal agreement and contracts. However, the

contracts that specify the relationship between the development team and customers and describe software business is still required [8].

4. **Responding to change over following a plan:** As the software development process is in progress, both the developer and the customers will gain more knowledge and a better understanding of the system, therefore the addition or the cancellation of some requirements may be necessary. The priority in the Agile manifesto is given to the response to the change in the development process life cycle rather than to follow a strict defined plan, since the ultimate goal to reach is the customer satisfaction.

In addition to these four values, the Agile Manifesto establishes twelve guiding principles for the Agile development-based methodologies that include [7]:

- **Principle 1**: "Our Highest Priority is to satisfy the Customer through Early and Continuous Delivery of Valuable Software."
  Early and continuous delivery of software will create a state of trust and flexibility between the customer and the development team; the development team will get a feedback on the correctness of their understanding of the product requirement through the customer meaningful feedback involvement, while the customers will feel more confident when they 'touch' a basic fully or partially functioning version of a component in their product, where they can prioritize an existing requirement or providing new ones.
- **Principle 2**: Welcome Changing Requirements, even late in Development. Agile Processes harness change for the Customer's Competitive Advantage.
  The corresponding implication of the impact of any embracing any change in requirement must be minimized through practices like 'refactoring' that will be discussed in the next sections.
- **Principle 3**: Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.
  This principle put some restriction on the first principle, it recommends the early and often delivery to satisfy the customer needs.
- **Principle 4**: Business People and Developers Must Work Together Daily throughout the Project.
  These frequent integrations aim to provide feedback and to answer the questions of the development team.
- **Principle 5**: Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
  The team members are the most vital factor in success, the Agile methods trust these individuals the power to make the appropriate decisions to do their job, for example, they can change the process steps if they thought it will be an obstacle to their team.
- **Principle 6**: The Most Efficient and Effective Method of Conveying Information to and within a Development Team is face-to-face Communications. This principle emphasizes the direct human communication in the agile team rather than using written specification, or written plans.
- **Principle 7**: Working Software is the Primary Measure of Progress. Breaking down the product into smaller pieces and the early and often delivery is a better progress measurement method that gives a more honest impact through the running codes.

- **Principle 8**: Agile Processes promote sustainable development: The sponsors, developers, and users should be able to maintain a constant pace indefinitely. Sustainable development means that the team should maintain a constant rhythm, i.e., the ultimate error free deliveries should not be the ultimate goal that must be met early by consuming all the available resources, instead the availability of delivering a high quality pieces and get the correct feedback is better until the final goal is reached eventually.
- **Principle 9**: Continuous Attention to Technical Excellence and Good Design enhances Agility. The highest quality code delivery is a must that should be always achieved to be agile, even if they have to refactor the code as a response to any change in requirement.
- **Principle 10**: Simplicity – the art of maximizing the amount of work not done – is essential. The aim is to produce a product that is simple and capable of handling the sudden changes and at the same time fulfill the customer requirements.
- **Principle 11**: The Best Architectures, Requirements, and Designs emerge from Self-Organizing Teams. Any agile team is a self-organized team that shares the responsibilities of a project and determine the best way to handle them through the knowledge they gain throughout the development process to achieve the optimal structure.
- **Principle 12**: At regular Intervals, the Team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

As the surrounding environment is subject to changes, the team must reformulate its structure, relationship and behavior based on it, i.e., at each iteration, the team continues the practices that served their needs and alter the ones that were obstacles in their way.

Due to its reduced costs and better productivity, quality, and satisfaction, the agile paradigm caused a large wave in the software development industry in its nearly two decades of age [11]. This is done with the help of its flexible handling techniques and improved communication and coordination mechanisms [12].

There is a misuse for the term of agile, agile is not a process neither a methodology, it is a set of practices, values, and principles. Agility is the ability to adaptively promote a quick response to any change, either in the environment, in the user requirements or in any delivery constraints. It is linked with nimbleness, suppleness, quickness, dexterity, liveliness, or alertness concepts [11]. The amount of agility the firm has will determine the degree of competitively it owns. Each time-boxed iteration in the agile framework includes planning, requirement analysis, and design, coding, and testing.

The main advantages that the agile based software development achieved [13] are:

- Improved communication and coordination among team members.
- The quick releases.
- The flexibility of design.
- More reasonable process.

One of the major characteristics of the agile development software is the ability to adapt to changes, as can be seen in figure 1, [14] stated that the conventional software

development processes respond to the change with nonlinear increasing cost as the project progresses. On the other hand, the incremental delivery property of agile process reduces this cost and flatten it, allowing a later stage change without a dramatic loss in cost and time.

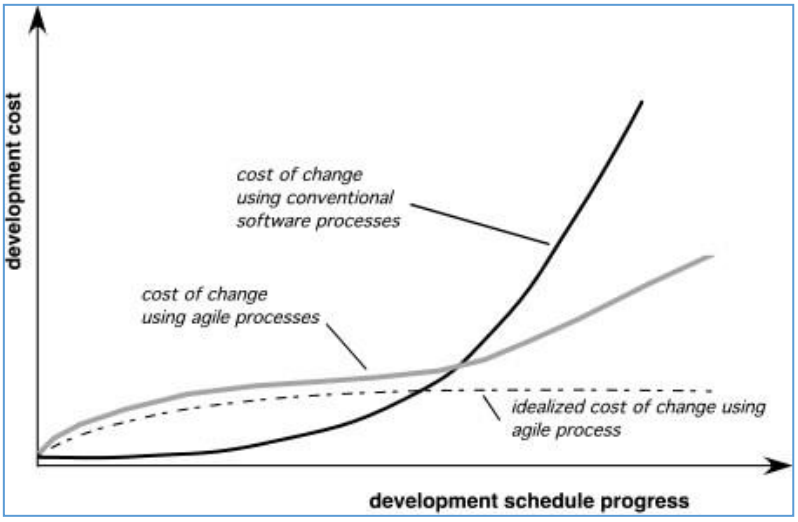

Fig. 1. The development cost vs. change in the development process [14]

3 Agile Software Development Versus Traditional Software Development

There are differences in some characteristics between the agile software development methods (lightweight) and the traditional software development methods (heavyweight), these characteristics can be categorized and presented in Table 1 [5], [15]:

Table 1. Agile Development vs Traditional Development

Parameter	Traditional Methods	Agile Methods
Ease of Modification	Hard	Easy
Development Approach	Predictive	Adaptive
Development Orientation	Process Oriented	Customer Oriented
Project Size	Large	Small or Medium
Planning Scale	Long Term	Short Term
Management Style	Command and Control	Leadership and Collaboration
Learning	Continuous Learning while Development	Learning is secondary to Development
Documentation	High	Low
Organization Type	High Revenue	Moderate and low Revenue
Organization's Number of Employees	Large	Small
Budget	High	Low
Number of Teams	Multiple	One
Team Size	Medium	Small

As shown in the previous comparison between the agile and the traditional software development methods; the agile methods are more adaptive to the requested changes, because of that, there is always a direct collaboration between the developers and the customers. Agile is more efficient to be used in small and medium projects. The planning in the agile methods is short term for specific functionalities which is easier to be done in comparing with the traditional methods which need a long-term planning or whole project planning. The documentation for the traditional methods must be high level, more detailed, so it needs heavy work and it will be a huge document. A number of teams and employees in the projects uses the traditional methods is more than the agile methods and needs higher budget amount [5], [15].

4 Agile Methods

Agile methods are processes that support the agile philosophy, i.e. agile values and principles. Each Agile method consists of a different combination of practices, which is a description of how the day-to-day work is done by the software developer. Each method differs from the other by choosing its appropriate set of terminology and practices [16].

There are different types of agile methods such as Test-Driven Development (TDD) method, Feature Driven Development (FDD) method, Extreme Programming (XP) method, Scrum method, Dynamic System Development Model (DSDM) method and Crystal methods etc. Each method has its own principles, life cycle, roles, advantages and disadvantages etc. All of these agile software development methods build the software in iterations and incremental processes. The comparison process will be covered as follows: we first list the factor of comparison; we introduce the case of this factor in Azure followed by its case in AWS, and we conclude each factor by a discussion that clarifies our findings.

4.1 Feature driven development (FDD) method

Feature Driven Development (FDD) method is one of the agile development methods, it manages short incremental iterations which leads to functional software. The feature is a valued function for the user in the required software. The idea of FDD was in 1997 by Jeff De Luca, he used this method in a large project when he realized that the other development methods are not useful to develop a large project in a specific time. The main idea about the FDD is to manage the software development based on the requirement feature list in the business needs. FDD is a high adaptive software development method that can accept late changes in the software requirements. The main focus of FDD is to deliver high quality outputs during all the phases of the development process [17].

FDD life cycle: FDD method life cycle contains five sequential processes as shown in figure 2, these processes are performed in an incremental iterative way where they will deliver the final software. These steps are [17], [18]:

a) Develop the overall model: In this step, the all the team members and the experts define the overall project context and scope which is required. Many models can be generated from different teams and experts, these models are reviewed and they select the most optimal model for the project based on the requirements.
b) Build the feature list: In this step the overall model and the requirement documentation are used to build the overall feature list for the system that the users' needs in the system. This feature list will be reviewed by the customer and the experts of the business then it will be confirmed.
c) Plan by feature: A high-level plan will be created in this step; this plan is derived from the previously approved feature list. The plan will be ordered based on the priority to the customer and the dependency between these features. This plan will contain a schedule for the major milestones of the project and a detailed schedule for each feature. All of the project manager, the development manager and the chief programmer are involved in this step. The chief programmer assigns the features to a specific developer who's called the class owner.
d) Design by feature: This step is an iterative step; each iteration can tack a few days but no more two weeks. Chief programmer and class owners produce a design package for each class, besides of the sequence diagrams. These design packages and diagrams are reviewed to be approved.
e) Build by feature: This the last step in the FDD process, in this phase the designs are implemented (coded), then the code will be inspected, do the testing processes. This step is also an iterative step same as the design by feature step, after all of the iterations are done then the developed features will be published in the main build, then a new feature set is started and so on.

Fig. 2. FDD Process [17]

FDD roles: There are six key roles in FDD method, each role can be done by more than one person, and one team member can do many roles. These roles are [17], [18]:

1. The Project Manager who is the leader of the project, provides the administrative decisions and the best working environment.
2. Chief Architect who is responsible for the overall design of the requested software and the final approval of the design is done by him.
3. Development Manager is the supervisor for all the development team, he has some good technical skills which is needed to solve some problems that the chief programmer cannot solve. He is responsible to solve conflicts between the teams.
4. Chief Programmer is the leader of the development process in the small teams, he leads the teams during all the analysis, design and implementation steps. He also selects the features which are needed to be done in this iteration and solve any implementation problem.
5. Class Owner is the person who is responsible for designing, coding, and testing the features.
6. Domain Expert is the expert in specific business, he has a clear understanding of how the business is working and what is the needs. He provides his knowledge about the business which will enrich the requested features list to be implemented.

Advantages and disadvantages of FDD: Using the Feature Driven Development methods has some advantages; these advantages can be summarized by [17]:

1. FDD is a high adaptive method and can accept late changes by the customer.
2. Deliver a high-quality result after each phase.
3. The results of each iteration can be delivered in one to four weeks; this will help to have quick feedback from the customers.

But there are some limitations and disadvantages of using the FDD such as [17]:

1. There is no guidance about the requirement gathering, analysis and the risk management in the FDD.
2. FDD requires an expert team with high skills in designing and modeling.
3. FDD does not concern about the criticality issues of the projects.

4.2 Test Driven Development (TDD) method

Another agile development method is called Test Driven development method (TDD), this method is based on building a small iteratively automated testing programs, then to write the code that can pass that test, and leave the enhancement of that code to

be done later. TDD is considered as the opposite of the traditional software development methods which do the test after the code is done. The idea of TDD was introduced in 2003 by Kent Beck, but it was already used by NASA in 1950 while they are working in the Mercury project. By following the TDD the fault and the bugs' rate will be decreased and will enhance the code quality. TDD has two main rules these are [17]:

- If the test fails, then write a code to solve it.
- Don't make duplications in the code.

The written tests by the developers will not be considered as the only needed test for the project, because of the developer write a single testing program that is needed in a small required functionality in the software. So, the software still needs to be tested using other types of tests such as the stress test, performance test etc. [17].

TDD life cycle: The TDD life cycle contains five main steps as discussed in figure 3, these set of steps will be done on each feature of the required software. After these steps are finished the resulted software will be a high-quality software, these steps are [17]:

1. Add a test; as we mentioned before, the first step in developing software using the TDD method is to prepare an automated test based on the requirement of the software. This automated test is written by the developer, he uses the customer requirements list and the use cases for these requirements.
2. Run all test cases on the code which is not available on this step, so all the test cases will fail to achieve some requirements.
3. Write the code, the developer will review the failed test cases and write the code that solves these test cases. The developer main objective in this stage is to pass the test, so the resulted code may not be efficient but it will be refined later.
4. Run all the test cases again, the developer will rerun all the test cases on the developed code, to check if the problem is solved or not. If the code is passed on all test cases, then the new code is satisfying the requirements needed. If not, then the code needs to be modified again and again until it will be passed.
5. Refactor the code, this step is the last step in the TDD, in this step the code will be refined, and removed the duplications. The code may be changed; but these changes must not affect the main objective of it, this means that the code must still pass the tests and the modifications do not effect on its main functionality.

Advantages and disadvantages of TDD: Using the Test-Driven Development methods has some advantages; these advantages can be summarized by [17]:

- Because of the testing is done while developing the software then the defects will be handled earlier and the bugs are located directly and no need for debugging. Other traditional software development methods the testing is done after the development is ended, so the fixing of the founded bugs will be a hard and complex process.
- The iterative testing will enhance the quality of the resulted code because all of the codes are tested while development.

- Dividing the overall software into small parts will make the development process easier, because the developer is concentrating on one problem to solve in each iterative.
- Refactoring the code will lead to better design and make it easier to be integrated with other functionalities.
- Developing small functionalities in each iterative will reduce the complexity of the overall software.

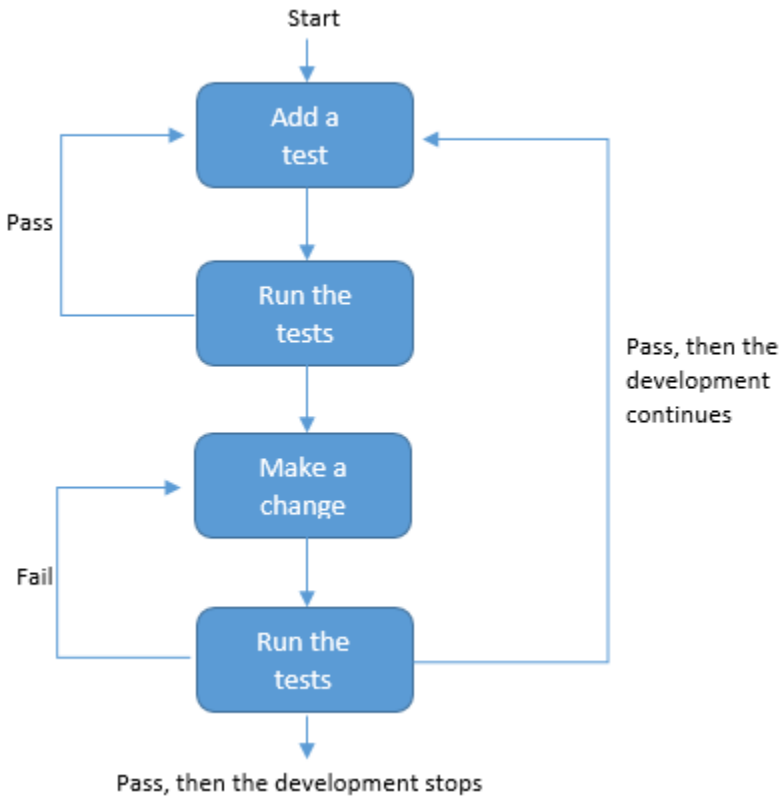

Fig. 3. TDD Process [17]

There are some disadvantages of using the test development methods, these are [17]:

- Programmers need some extra skill which is writing the testing cases which is the duties of the testers, so the programmers will be confused.
- Hard to use the TDD methods in the projects which contain synchronizations between its functionalities and parts.
- TDD leads to poor documentation because the TDD only uses the test cases for the development, where the documentation can be used in the maintenance phase.
- TDD sometimes consumes time when the repeated failures happened.

- TDD is not guided by management principles in software development, it only focuses on the development activities.

4.3 Dynamic System Development Method (DSDM)

Another agile software development method is called Dynamic System Development Method or Model (DSDM), this method uses the rapid application development approach (RAD). It is an incremental iterative approach where the quality of the software is very critical value in this method. The idea of DSDM was in 1994 by practitioners of a consortium in the United Kingdom, then it became a framework for the rapid applications in 1997. The main idea about the DSDM is to set and define the resources of the project and the time for the project and then adjust the amount of functionality that can be done in these time and resources. It is the opposite of the traditional methods where the list of functionalities required in the system are predefined first then the time and the resources will be allocated and defined based on these resources. [17], [18].

DSDM life cycle: Dynamic System Development Method contains five main phases. DSDM as we discussed before it is an incremental method; so, the feedback of clients is directed processed to improve the quality of the software. DSDM phases are: [17], [18].

1. Feasibility study phase which is the first step in the DSDM method, in this step the project will be studied to decide if the DSDM is appropriate to use in the development or not. Also, the risk analysis and specifying the technical requirements are done in this phase. The feasibility report and the outline plan for development are the results from this step.
2. Business study phase, in this phase meetings are done between the business experts from the client side and the development team, these meetings are done to specify the user's requirements and to make a list of functionalities needed in the system with its priorities. Technology which will be used is specified in this step, also a high-level description of the process, ER-Diagrams, object mode, system architecture, system architecture and the outline prototyping plan are the results of this step.
3. Functional model iteration, in this phase the analysis, coding and prototyping are done iteratively and incrementally, the porotypes are analyzed to enhance the analysis model. The main objective of this step is to specify what will be developed in the software, when it will be developed and the development way. Also, the prioritized functions, functional prototyping, nonfunctional requirements and the risk analysis.
4. Design and build phase, this phase is also an iterative phase, in this phase, the identified requirements from the previous phase is implemented and coded, and then it will be published and tested by the users. The user's feedback is used to enhance the system iteratively. So, a tested software with an at least minimal set of the requirement is the result of this step.
5. Implementation phase, in this phase the software moved from the development to the production and it will be handled to the users, training sessions are provided. Also, the

user manual is developed and handled. If the software is fulfilling all the user's requirements, then there are no other development processes is needed. If not, the whole process is redone again.

DSDM roles: There are many different roles used in dynamic software development method, the primary roles are as following [17], [18]:

- The developers who are all the development team including the senior developer, analyst, designer, programmer and the tester.
- Technical coordinator who is the person who ensures that the business and the technical aspects of the project are following the plan. He defines the architecture of the system and ensures the technical quality of the project.
- Ambassador user who is from the customer side, he is the person who will use the developed system, he should be able to discuss all the users' needs to the development team.
- Advisor user who is also from the customer's side, he can provide some important viewpoints that is needed in the project where the ambassador user can't provide.
- Visionary who is the user who has a good understanding of the whole business needs and objectives. He assures that the most important business requirements are provided to the developers and are done in the right way.
- Executive sponsor is the person from the customer side who have the power of making any decision and has the authority on the financial issues.

DSDM advantages and disadvantages: There are many advantages of using the dynamic software development method, such as [17], it provides rapid application development based on the agile principles (focus on the business needs, deliver on time, focus on the quality, incremental and iterative development, direct feedback). DSDM can be incorporate best practices from other approves. It can provide guidelines for the project aspects such as project management, risk control, and development techniques. However, there are some disadvantages of using the DSDM [17], there are a large number of roles in the DSDM, this will lead to some difficulties in the administration of the project, DSDM does not consider the project criticality, and DSDM does not provide specific guidelines for the issues related to the team size and the length of the iterations.

4.4 Scrum method

Scrum is a concrete implementation of an agile framework that was proposed in [20] for project management for the iterative software development process. It focuses on delivering the highest value in the shortest time. It is a team oriented agile methodology that specifies a certain role, establishes a short time boxed iteration called sprints in which the system is incrementally developed and produces a different artifact that coordinates its work [19]. It is considered among the most used agile methods [8]. This popularity is due to its simplicity and due to it focusing on the software management issues rather than the technical software development practices which make it widely applicable to any domain.

SCRUM life cycle: There are three main phases in Scrum [8, 18] that are described below and illustrated in figure 4:

1. The initial phase is the outline planning phase or the pre-sprint phase [18] where the general objectives for the system being developed and designed are outlined. Also, the definition of the project team, required tools and resources are stated. It is sometimes called iteration 0. [21]

    - A Product Backlog is generated and used for documenting the customer requirements as user stories and features [22]. User stories are often defined according to the pattern [8]: 'As a <User>, I want to<Have>so that <Benefit>'
    - The requirements are then analyzed and given specific priorities and the implementation effort estimation is carried by the product owner whose responsibility is to maintain a visible and transparent product backlog.
    - The product Backlog is subject for continuous updating since the user stories are created incrementally, and also throughout the development process the priorities of the existed stories may change.

2. The development phase or the sprint phase: It consists of a series of sprint cycles, where the output of each cycle is an increment value that is added to the system. Below are the details of a sprint cycle:

    - Sprints are iterative fixed length cycles with length from 2–4 weeks.
    - Each sprint goes through the traditional software process phases that start from requirement analysis or sprint planning from the product backlog, through the design and ending up in the delivery phase after the sprint review.

In the sprint planning meeting that is held at the beginning of each sprint that combines both the development team and the product owner, an agreement on the work that must be done during the next sprint is made. The user stories are incrementally transferred from the product backlog to the sprint backlog based on their priorities. Furthermore, the features and functionalities that should be implemented are decided. Upon the completion of the pre-sprint planning, a sprint cycle begins. During a Sprint, the team is isolated from the external distractions by the Scrum master, and the features are implemented and tested daily through a meeting that is called daily scrum. Daily scrum meetings are 15-minutes-long and conducted to enhance the communication, to synchronize the activities, to reconcentrate the team focus on the common goal that is shared by all team members, and to solve any problems or obstacles that may face the team. At the end of the sprint, two meeting are held; the sprint review meeting in which the sprint output "a potentially shippable increment" is inspected, analyzed and assessed, and the sprint retrospective meeting to discuss possible future improvement [19].

- Burn down chart is used to monitor the status of the sprint.
- There may be more than one team that build the increments, i.e., parallel sprints.

3. The project closure phase: This phase is when the requirements achieved and the goals required are identical based in the agreement of the product owner and the team. The latest version of the product is now ready for the release and ready for the distribution. In addition to the readiness of complete documentation and user manuals.

SCRUM roles: There are three main role players in the scrum methodology:

1. The product owner: Is the responsible for defining the development teams' targets by creating and prioritizing the customer requirements according to the market value and converting them into the required features as user stories in the Product Backlog. Moreover, the product owner reviews the increment outcome of the system of each sprint.
2. Scrum master: Is a project manager that is responsible for forcing and monitoring the scrum values and rules in the project. The scrum master communicates with customers and management outside of the team, conducts the Scrum Meetings, and protects the sprint from any outside interference during the operation. The scrum master also measures the progress against the backlog and ensures that the team is fully optimized and functional by eliminating any impediments that may face the team and provide them the necessary resources to keep them producible. However, the scrum master does not organize and divide the responsibilities to the team.
3. The development team is a self-organized team that is committed and collaborate with each other to achieve the sprint goals and to deliver a potentially releasable product increment at the end of each sprint. Their responsibilities include analyzing the requirements and design, developing, testing and validating the produced software. The team must be cross functional, i.e., it has the necessary and needed skills to complete the job.

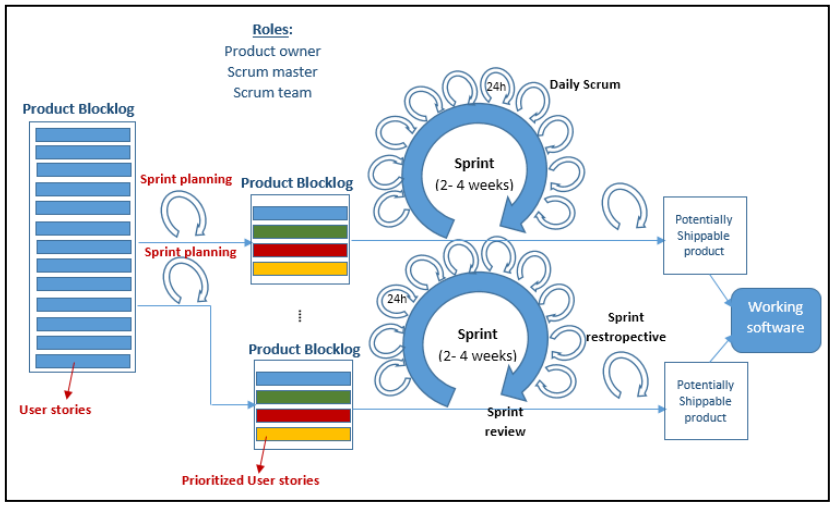

Fig. 4. Scrum method life cycle

Advantages and disadvantages of SCRUM: The scrum method is the most powerful method that have many advantages that can be summarized by [8, 18]:

- The software product is divided into a smaller set of manageable and understandable components shared among the teams; thus, this increase the communication and the shared knowledge [23].
- Transparency: The team has visibility of everything including the communication and the feedback from the product owner through the different meetings that are conducted through the development process.
- Self-organization: All the team shares the responsibilities.
- Self-retrospective: Provide a self-assessment tool of achieved goals versus the required ones after each iteration or sprints, this increase productivity through continuous testing.
- Simple process.
- Ignoring any change within the sprint duration by forbidding any feature to be added to the sprint which allows the team to complete their current in-progress functionalities.

However, as there is no fully optimal method here are some of the scrum disadvantages:

- Some violation of responsibility may occur since there exist no precisely defined responsibilities for each team member.
- Scrum does not prescribe any specific practices or working methods or any guidance on the engineering practices [21].

4.5 Extreme programming method

Extreme programming (aka XP) is one of the early agile methods that was proposed by Kent Beck.

[24] in 1999 to overcome the limitation of the convolutional software development process in the front of speedily frequent changing requirements, and to develop a methodology that is suitable for object-oriented project that consists of multiple programmers in a single location [13]. It is an integration of well-known software engineering practices. It attempts to reduce the cost of the requirements changes by replacing the long development cycle by multiple short cycles [13] to achieve a customer satisfaction. XP aims to improve the software quality by taking the concepts of software engineering to an extreme level [19].

XP has 13 primary technical practices which are: sit together, whole team, informative workspace, energized work, pair programming, stories, short iteration, quarterly release, slack, ten-minute build, continuous integration, acceptance- and unit test-driven development, and incremental design, in an addition of 11 corollary practices: root cause analysis, collective code ownership, code and tests, negotiated scope contract, real customer involvement, incremental deployment, team continuity, shrinking team, daily deployment, single code base, pay-per-use practices[21]. As a result of these practices, there are five key values of XP: communication, simplicity, feedback, courage, and quality work [23].

XP life cycle: The life cycle of XP consists of six phases [18] that can be depicted in figure 5 and explained below:

1. The Exploration phase:

    - The customer is an essential part in the XP team, he is responsible for making decisions about the requirements and features that are expressed as user stories that they think must exist in the first release.
    - The project team gets introduced to the available resources such as the tools, technology and practices in order to be familiar with them in the project.
    - A sample prototype of the system is built to test the technology and to discover the possible architecture of the system.
    - Duration: A few weeks to a few months.

2. The Planning phase:

    - An effort estimation and the schedule are made and agreed upon by the programmers.
    - The stories are prioritized.
    - Duration: A couple of days.

3. The Iterations to Release phase:

    - It includes several iterations with each iteration lasts from one up to four weeks.
    - The first iteration is a special iteration with the goal of creating an overall system architecture by selecting the appropriate stories.
    - The functional tests are developed by the customer and ran at the end of every iteration.
    - The last iteration output will be ready for a production system.

    Two main practices in this phase: pair programming and refactoring: Pair programming and Refactoring; Pair programming, in which the developers work in pairs to develop the code, i.e., one is actively writing the code, and the other observes, supports and reviews that code [19]. This has many benefits such as spreading the knowledge across the team especially that these pairs are selected dynamically, and developing a collective ownership and common responsibility of the developed code. Moreover, it supports refactoring to improve the software. Refactoring is a tool to improve the software and makes it simple and maintainable by reconstructing its code without changing its functionality [19], for example by removing duplication, adding flexibility [18], or renaming the code variables and functions for a better understanding.

4. The Productionizing phase:

    - An additional testing and evaluating of the performance of the system are needed before it can be released to the customer.

- After the first release of the system is delivered to the customer, the system should be kept running while new iterations are produced.

5. The Maintenance phase: incorporating new people into the team may be required to support customer tasks.
6. The Death phase: In this phase, no additional user stories existed, thus no more changes are requested and the current system implementation satisfy the user requirements and needs. Furthermore, Death may occur if the system stops delivering the desired results, or if any further development is costly.

XP roles: There are different roles in XP for different tasks [18]:

- **Programmer**: Is responsible for writing simple, high quality functioning codes, usually done by the collaboration with other programmers according to the pair programming practice.
- **Customer**: The customer writes the system requirements and features as stories and functional tests, and assign them the appropriate priorities and decides at the end of the process if each requirement is satisfied or not.
- **Tester**: Testers assist the customer in writing the functional required tests and run the tests.
- **Tracker**: Tracker analyze the team estimations and progress in each iteration and provide them with his feedback.
- **Coach**: Is responsible for guiding the overall process.
- **Consultant**: Consultant is qualified outside the team member that help the team in solving any encountered problems.
- **Manager**: A manager is capable of making decisions and responsible for the communication and elimination of any obstacle in the team.

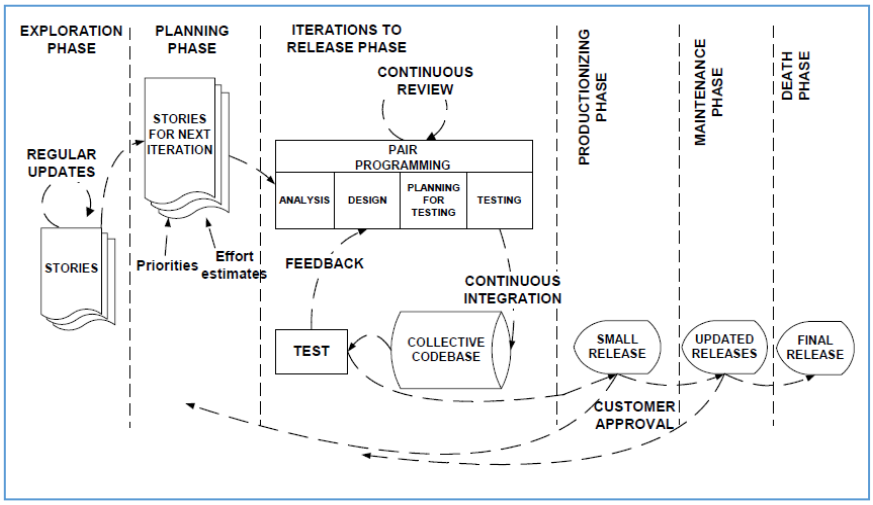

Fig. 5. Extreme Programming life cycle [18].

Advantages and disadvantages of XP: XP has many advantages [18], some of them is listed below:

- Incremental development is supported through small, frequent system releases.
- Improve the productivity; the rapid feedback through the “Extreme” levels of practicing leads to more responsive software, i.e., many versions can be built each day and only accepted if they pass the testing [8].
- Maintaining simplicity through constant refactoring of code.
- Improve the Quality through the development of automated tests before integrating a feature in the system [8].

On the other hand, the XP suffers from the following limitations [18]:

- It lacks the capability to support the distributed teams since it focuses on the community and co-location.
- The test-driven development practice requires additional technical training for associated team members.
- XP depends on the informal documentation; code, story cards, etc. to keep the important details of the project. Therefore, XP must be used in the projects that require traceability and audit-ability [21].
- The real customer involvement practice is effective but stressful, and time-consuming [21].

5 Big Data Systems Using Agile Development

The term Big Data can be defined as "massive data files that have a large, diverse and complex structure and are therefore difficult to store, analyze and visualize for other processes and results" [26]. There are three main features of big data referred to as 3V'S; Velocity, Volume, and Variety. Volume feature refers to the amount of large data generated from many different sources in a period of time. Velocity term refers to the speed at which the data is generated or analyzed. Variety refers to the many and different sources and types of data both structured and unstructured such as text, images, audio and video [27], [28]. The development of Big data system differs from the development of traditional small structured small data systems and this is due to several challenges. First, technical challenges to develop big data system depend on building architecture for integrating, processing, storing, and analyzing different types of data; structured, semi-structured and unstructured. The second challenge is how data scientist can work together with software engineer determine and maximize the value of big data [29].

A number of papers have been published on applying agile methodologies to big data. Franková et al. [26], worked on a study to clarify the big data management and the possibility of applying the agile approach to big data projects. They collected data by interviewing experts in the project and big data management in different. Data is evaluated to identify which agile manifesto principles can be used in big data projects management. They found that the suitability of agile and plan driven approach is determined according to not only by the size and seriousness of the project, but also by the

dynamic environment, the specialty of team that is working on the project and organizational culture. Besides, they recommended to apply an agile approach in big data management.

Larson and Change [9], analyzed agile methodologies that could be applied to business intelligence delivery and discussed the best practices of agile business intelligence considering the impact of big data. They proposed an agile framework for Business intelligence delivery which addresses the influence of big data on business intelligence. This framework consists of two layers; business intelligence delivery and fast analytics-data science. The business intelligence delivery layer is the top layer of the framework, which includes five steps: discovery, design, development, deploy and value delivery. These steps are sequential steps that contain specific tasks that work to achieve business objectives and goals. The second layer is fast analytics and data science layer, it is the bottom of six steps: scope, data acquisition/discover, analyze/visualize, validate and deployment, these two layers integrated and work together to ensure the execution and management for the framework.

Chen et al. [29] presented an Architecture-centric Agile Big Data Analytics (AABA) development methodology, they revealed that architecture agility is the key for successful agile big data analytics development. This architecture allows all stakeholders to communicate tightly to determine the value of the proposition for the system being built and to focus on the important tasks such as value validation. It also provides a basis for value discovery with stakeholders, and for planning and estimating cost and schedule. Dharmapal et al. [30], presented that there are three phases of big data analytics using the agile model, these phases are planning Phase, development phase, and closure phase. Planning phase involves all stakeholders are identified and the requirements are written by product owner as a story. These stories are arranged according to priority in a way that provides the functionality that can be tested and delivered independently. The data is collected incrementally based on needs. These data are analyzed at the development stage and the requirements are developed in a repetitive manner to achieve the final objectives and goals. The closure phase occurs when all requirements are met, and final testing and implementation will take place.

6 Cloud Computing and Agile Development

Cloud computing relies on the Internet where services, applications, and infrastructure are accessible online. With the growing maturity of cloud computing technologies, there are many research interests in using agile development software to build cloud applications. While Cloud services fulfill user requirements quickly, Agile methodologies tend to separate project requirements into smaller, achievable parts. This policy ensures user feedback information on each task of the project. These parts can be planned, developed, and individually tested to maintain a high standard quality and avoid various problems that may occur [32].

The integration of cloud application development with agile methodologies offers many advantages in reducing time and cost, increasing software quality, and effective

use of resources [31]. Furthermore, cloud computing eliminates communication problems among project participants by using much software as service solutions that help communicate and collaborate among project participants when sharing resources [32].

Cloud agile framework for cloud application development is proposed by Das and Vaidya [33]. It will be based on agile CSC which is used to deliver increments of high value iteratively. There are three roles in the framework, product owner who arranged the requirements into priorities, scrum master who own the processes, and a team who work together to fulfill the project. The framework involves four phases; inception, elaboration, construction, and construction. In the inception phase, the project will be set and initial set of requirements will be collected and prioritized, and the architecture will be discovered to pool the resources and services. The goal of elaboration phase is to ensure that the team is able to work together to produce an initial, incremental working application. In the construction phase, the application is completed waiting for user acceptance. User acceptance and application delivery is the objective of the construction phase.

In the review study of Younas et al. [34], they focused on techniques employed in cloud computing environment that are useful for agile development, and discussed the current state of cloud computing using agile software development practices the impact of cloud computing on agile development, and cloud challenges and benefits when adopting in agile software development. They also classified the type of solutions for practicing agile development in a cloud computing environment. They concluded that more empirical studies are needed in agile software development and cloud computing and cloud computing has a positive impact on agile software development. The challenges due to cloud computing are security concerns, lack of practical experience, and safety of the development environment.

Butt [35], presented an overview of agile methodologies with cloud computing, and made comparisons between agile methodologies based on the following characteristics: development approach, iteration time period, project team, team communication, customer involvement, project documentation, specialties, quality, and time. Cloud deployment models are also discussed and the benefits of agile methods of integration with cloud computing. For future research, the recommendations focused on the usefulness of combining cloud computing with agile development.

7 Coordination and Agile Development

Software development is a cooperative work, tasks may be distributed over different teams, these tasks must be managed and ordered based on some criteria. Tasks can be done in parallel, but there are some tasks cannot be started before another task is ended. So, the need for coordination between these tasks, processes and teams is a must to gain the best software or product with the minimal cost [3]. There are many objectives of coordinating the execution of dependent processes such as optimal use of limited shared resources, managing producer-consumer relationships, Managing Simultaneity constraints, and managing task-subtask dependencies. Because of the importance of serving the coordination between the tasks and the processes, several studies have been

conducted to describe the dependencies between these tasks [36-40]. Some provide an approach or model of coordination in the agile software development, define agile practices that have a coordinating function and explain how they fit together to form a coordination strategy [36], [37]. Other studies identify the coordination mechanisms that can facilitate the development of large-scale agile development such as reference [38]. Strode et al [36], presented a coordination model based on a study of co-located agile development teams. The coordination strategy components are synchronization, structure, and boundary spanning. Synchronization is Activities performed by all team members simultaneously such as weekly meetings. Structure involves proximity which is the closeness of individual team members, availability of team members and substitutability which maintains time schedule between team members. Boundary spanning which involves activities, artifact, and coordination role that is taken by a team member. Inter-team coordination in large-scale globally distributed scrum is reported by Paasivaara et.al [40], after addressing two case studies, they found that teams that followed the agile practice of coordinating through Scrum of Scrums were not able to coordinate effectively.

8 Conclusion

In this research we explained many types and methodologies for agile software development, each methodology has its own advantages and disadvantages, so there is no optimal methodology for all types of projects, each project has its own specifications, characteristics and needs to be done. Therefore, selecting the best agile methodology to be used in the project development must be done carefully based on these variabilities. Or sometimes there is no agile methodology can be used on some projects development so the traditional methods can be optimal for these cases, such as the organizations with a large number of teams and employees, and projects with a critical huge budget. Agile methodologies can be used in organizations with a small number of employees, low budgets projects.

At the beginning of the agile revolution, XP was the most popular agile method, however, since 2004 the interest in XP as a methodology for applying agility is declined. Nevertheless, its developmental practices are still very powerful and used such as pair programming, test-driven development, and user stories. Scrum is a more general methodology that concentrates on project management; therefore it leads the agile development trends nowadays and plays a very influential role. The TDD keep a stable research interest since 2007. Furthermore, in this research, we discussed how can agile be implemented in big data systems and a cloud computing environment.

9 References

[1] Braude, E. J., & Bernstein, M. E. (2016). Software engineering: modern approaches. Waveland Press.

[2] Iee, E. (1990). IEEE standard glossary of software engineering terminology.

[3] Giuffrida, R., & Dittrich, Y. (2015). "A conceptual framework to study the role of communication through social software for coordination in globally-distributed software teams". Information and Software Technology, 63, 11-30. https://doi.org/10.1016/j.infsof.2015.02.013
[4] Mall, R. (2018). Fundamentals of software engineering. PHI Learning Pvt. Ltd.
[5] Matharu, G. S., Mishra, A., Singh, H., & Upadhyay, P. (2015). "Empirical study of agile software development methodologies: A comparative analysis". ACM SIGSOFT Software Engineering Notes, 40(1), 1-6. https://doi.org/10.1145/2693208.2693233
[6] Jammalamadaka, K., & Krishna, V. R. (2013). "Agile software development and challenges". International Journal of Emerging Technology and Advanced Engineering, 3(6).
[7] Conboy, K. (2009). "Agility from first principles: Reconstructing the concept of agility in information systems development". Information systems research, 20(3), 329-354. https://doi.org/10.1287/isre.1090.0236
[8] Rodríguez, P., Mäntylä, M., Oivo, M., Lwakatare, L. E., Seppänen, P., & Kuvaja, P. (2018). "Advances in Using Agile and Lean Processes for Software Development". Advances in Computers (Vol. 113, pp. 135-224). Elsevier. https://doi.org/10.1016/bs.adcom.2018.03.014
[9] Larson, D., & Chang, V. (2016). "A review and future direction of agile, business intelligence, analytics and data science". International Journal of Information Management, 36(5), 700-710. https://doi.org/10.1016/j.ijinfomgt.2016.04.013
[10] Gupta, S., & Gouttam, D. (2017)." Towards changing the paradigm of software development in software industries: An emergence of agile software development". 2017 IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM) (pp. 18-21). IEEE. https://doi.org/10.1109/icstm.2017.8089120
[11] Erickson, J., Lyytinen, K., & Siau, K. (2005). "Agile modeling, agile software development, and extreme programming: the state of research". Journal of Database Management (JDM), 16(4), 88-100. https://doi.org/10.4018/jdm.2005100105
[12] Rauf, A., & AlGhafees, M. (2015). "Gap Analysis between State of Practice and State of Art Practices in Agile Software Development". In 2015 Agile Conference (pp. 102-106). IEEE. https://doi.org/10.1109/agile.2015.21
[13] Choudhary, B., & Rakesh, S. K. (2016). "An approach using agile method for software development". 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH) (pp. 155-158). IEEE. https://doi.org/10.1109/icicccs.2016.7542304
[14] Pressman, R. S. (2005). Software engineering: a practitioner's approach. Palgrave Macmillan.
[15] Vijayasarathy, L. R., & Butler, C. W. (2016). "Choice of software development methodologies: Do organizational, project, and team characteristics matter?". IEEE software, 33(5), 86-94. https://doi.org/10.1109/ms.2015.26
[16] Elbanna, A., & Sarker, S. (2016). "The risks of agile software development: Learning from Adopters". IEEE Software, 33(5), 72-79. https://doi.org/10.1109/ms.2015.150
[17] Anwer, F., Aftab, S., Waheed, U., & Muhammad, S. S. (2017)." Agile Software Development Models TDD, FDD, DSDM, and Crystal Methods: A Survey". International journal of multidisciplinary sciences and engineering", 8(2), 1-10.
[18] Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017)." Agile software development methods: Review and analysis ". arXiv preprint arXiv:1709.08439.
[19] Schmidt, C. (2016). Agile software development teams. Springer International Publishing.
[20] Schwaber, K., & Beedle, M. (2002). Agile software development with Scrum (Vol. 1). Upper Saddle River: Prentice Hall.

[21] Williams, L. (2010). "Agile software development methodologies and practices". Advances in Computers (Vol. 80, pp. 1-44). Elsevier.
[22] Chopade, M. R. M., & Dhavase, N. S. (2017). "Agile software development: Positive and negative user stories". 2nd International Conference for Convergence in Technology (I2CT) (pp. 297-299). IEEE. https://doi.org/10.1109/i2ct.2017.8226139
[23] Cohen, D., Lindvall, M., & Costa, P. (2003). Agile software development. DACS SOAR Report, 11, 2003.
[24] Beck, K., & Gamma, E. (2000). Extreme programming explained: embrace change. addison-wesley professional.
[25] Jeffries, R., Anderson, A., & Hendrickson, C. (2001). Extreme programming installed. Addison-Wesley Professional.
[26] Franková, P., Drahošová, M., & Balco, P. (2016). "Agile project management approach and its use in big data management". Procedia Computer Science, 83, 576-583. https://doi.org/10.1016/j.procs.2016.04.272
[27] Sagiroglu, S., & Sinanc, D. (2013)." Big data: A review". 2013 International Conference on Collaboration Technologies and Systems (CTS) (pp. 42-47). IEEE. https://doi.org/10.1109/cts.2013.6567202
[28] Al-Saqqa, S., Al-Naymat, G., & Awajan, A. (2018). "A Large-Scale Sentiment Data Classification for Online Reviews under Apache Spark". Procedia Computer Science, 141, 183-189. https://doi.org/10.1016/j.procs.2018.10.166
[29] Chen, H. M., Kazman, R., & Haziyev, S. (2016). "Agile big data analytics development: An architecture-centric approach". 49th Hawaii International Conference on System Sciences (HICSS) (pp. 5378-5387). IEEE. https://doi.org/10.1109/hicss.2016.665
[30] Dharmapal, S. R., & Sikamani, K. T. (2016)." Big data analytics using agile model ". 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) (pp. 1088-1091). IEEE. https://doi.org/10.1109/iceeot.2016.7754854
[31] Almudarra, F., & Qureshi, B. (2015). "Issues in adopting agile development principles for mobile cloud computing applications". Procedia Computer Science, 52, 1133-1140. https://doi.org/10.1016/j.procs.2015.05.131
[32] Kalem, S., Donko, D., & Boskovic, D. (2013)." Agile methods for cloud computing". In 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 1079-1083). IEEE.
[33] Das, D., & Vaidya, K. (2011)." An agile process framework for cloud application development". Computer Science Corporation, Falls Church.
[34] Younas, M., Jawawi, D. N., Ghani, I., Fries, T., & Kazmi, R. (2018). "Agile development in the cloud computing environment: A systematic review". Information and Software Technology. https://doi.org/10.1016/j.infsof.2018.06.014
[35] Butt, S. A. (2016). Study of agile methodology with the cloud. Pacific Science Review B: Humanities and Social Sciences, 2(1), 22-28. https://doi.org/10.1016/j.psrb.2016.09.007
[36] Strode, D. E., Huff, S. L., Hope, B., & Link, S. (2012)." Coordination in co-located agile software development projects". Journal of Systems and Software, 85(6), 1222-1238. https://doi.org/10.1016/j.jss.2012.02.017
[37] Scheerer, A., & Kude, T. (2014). "Coordination in large-scale agile software development: A multiteam systems perspective". In 2014 47th Hawaii international conference on system sciences (pp. 4780-4788). IEEE. https://doi.org/10.1109/hicss.2014.587
[38] Nyrud, H., & Stray, V. (2017). "Inter-team coordination mechanisms in large-scale agile". In Proceedings of the XP2017 Scientific Workshops (p. 16). ACM. https://doi.org/10.1145/3120459.3120476

[39] Dingsøyr, T., Bjørnson, F. O., Moe, N. B., Rolland, K., & Seim, E. A. (2018). "Rethinking coordination in large-scale software development". In Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering (pp. 91-92). ACM. https://doi.org/10.1145/3195836.3195850

[40] Paasivaara, M., Lassenius, C., & Heikkilä, V. T. (2012). "Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work?" In Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement (pp. 235-238). ACM. https://doi.org/10.1145/2372251.2372294

10 Authors

Samar Al-Saqqa is currently a teacher with the University of Jordan, King Abdullah II School for Information Technology, Information Technology Department, holds a M.Sc. and B.Sc. in Computer Science from The University of Jordan, King Abdullah II School for Information Technology, Information Technology Department. The research interests in the areas of Natural language processing, Sentiment Analysis, Machine Learning, Data Mining and e-technologies. (Email: s.alsaqqa@ju.edu.jo).

Samer Sawalha received the B.Sc and the M.Sc degrees in Computer Science from Princess Sumaya University for Technology (PSUT) Amman - Jordan; he is a PhD student in the same university (King Hussein School of Computing Sciences) in Computer Science. He is working as a Senior System Analyst in the Royal Scientific Society Amman – Jordan since 2008. Blank Background Image Compression and Storage was the title of his thesis for the master degree he got. His research interests are in IoT, Image Compression, Data Storage, Database Management and Storage. (Email: samerfs@hotmail.com)

Heba Abdel-Nabi, a PhD student in Computer Science at Princess Sumaya University for Technology (PSUT). She received her Bachelor degree in computer engineering in 2010 and master degree in electrical engineering in 2015 from PSUT. Her research interests are digital image processing and information security, deep learning, artificial intelligence and evolutionary algorithms. (Email: h.yousif88@yahoo.com)

Article submitted 2020-01-20. Resubmitted 2020-03-04. Final acceptance 2020-03-29. Final version published as submitted by the authors