Journal of Theoretical and Applied Information Technology
30th November 2019. Vol.97. No 22
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

THE ANALYSIS OF CURRENT STATE OF AGILE SOFTWARE DEVELOPMENT

SAMER ATAWNEH

Saudi Electronic University, PO Box: 93499, 11673, Riyadh, Saudi Arabia
E-mail: satawneh@seu.edu.sa

ABSTRACT

The agile software development methods are studied in this paper. Agile software development methodology was formally represented to the community of software engineering through twelve principles and four core values. Agility is considered the cornerstone of the agile software development. This contrasts with the plan-driven technique that is explained in different conventional models (e.g. Waterfall). Currently, the agile development is an important development approach, which is derived from practical uses to encourage the cooperation between users and developers so that fast development processes could be supported, and to adapt with the modifications that are affecting the dynamic environment. Many agile methods are currently available in the literature with Scrum and Extreme Programming (XP) methods forming two most commonly used methods. This study demonstrates the value of applying the agile methods in developing software projects by analysing the current agile methods. The study results reveal that the agile development introduces significant benefits over conventional methodologies. However, these benefits are not compatible with all projects and situations. The results also show a decline in the interest in XP, while the interest in Scrum is increasing all the time.

Keywords: *Agile development, XP, Scrum, Adaptive software development, Crystal, Lean development*

1. INTRODUCTION

The aim of this introduction is to portray the meanings that are recently correlated to "agile", to give a definition of the agile development. Agile development methodology (also known as lightweight development methodology) is a software development framework that relies on pre-existing incremental and iterative development principles [1] (for example, Spiral and Waterfall methodologies). This methodology uses continuous planning, development, and testing and continuous contact with system stakeholders [2]. Most software development organizations are gravitating and moving towards agile software development methods [3]. With the mass movement towards this methodology, the software development using agility is becoming the mainstream [4]. Note that agility is a cornerstone of the agile software development [5]**.** Conboy [6] formally defined agile development as ''the continual readiness of an information systems development method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value, thereby bringing about quality, economic benefits, and simplicity values achieved through relationships with its environment and by its collective components."

The general agile framework approach is that short development cycles are involved in which a flexible approach is used in software product development [1, 2]. This approach allows teams to be self-managing and adaptive to change in requirements, where priority is based on the ever-changing requirements. In addition, one major feature of agile development is team interaction, collaboration and collective decision-making. The Agile framework also emphasizes stage-by-stage of a software product delivery providing fine-tuning or additional validation of the software's feature set with each delivery phase. With this approach, the usual excessive negotiations in a development project (when scope is modified or system requirement changes) is minimized or totally eliminated.

Agile development approaches articulate and describe all required processes, enumerate deliverables, assigns roles for defining specifications, designing, implementing and verifying a software product. Over the last decade, the development processes are constantly changing and evolving. With the surging popularity of agility as an approach for developing software projects, the general approach used by software development companies in managing development teams has radically changed. In late 1990s, varied methods of agile methodology such as Scrum, XP, Crystal, lean software development, and Feature-driven Development (FDD) emerged and attained popularity, as they all attempted to address the core Agile manifesto principles [7, 8]. The Agile manifesto [1] will be highlighted in Section 2.2.

1.1. Software Development Process

One of the major overarching focus and goal of the software engineering discipline is to offer appropriate solutions to existing practical software development problems [3, 9]. Currently, the software engineering discipline has matured into independent profession and a domain that is tightly associated with computer sciences and other conventional engineering disciplines. Overall, software engineering discipline handles all processes necessary to solve real software development problems in an optimal and reliable way. Software engineering principles equip and empower software developers with disciplined, quantifiable methods and tools that guide them with best-practice guidelines and systematic approach to software design, implementation, post implementation operation, and maintenance" [10]. Software engineering discipline also encompasses applying several theories, processes, methods and utilization of tested tools to develop and maintain software systems in concert with the needs of organizational software development constraints.

Developing software systems is considered to be a complicated process where several software projects end in overdue results of these projects. For instance, failure to achieve the goals of the project, or results containing project annulments [11]. Such an increased failure level is unanticipated when the history of over 60 years is derived for researches regarding project failures, several best-practices books, software development projects, countless development tools, processes and methods. There are many complex sources that make increased failure levels of software projects more obvious. The main grounds behind that refers back to the essential characteristics of a software, the used software contexts, the tasks complexity pertaining to the software development, and the overall nature of the software projects.

The software development process is defined as "a set of activities that lead to produce a software product" [9]. Since the improvement of various software might need several processes, extremely different processes are improved by the software engineering discipline over the past decades ago. Nonetheless, there exist a number of tasks in which each improved project should involve. For instance, the tasks of implementation for designing, coding and testing the software, the definition tasks for requirements specification, and the tasks of evolution for corrections and adaptations. Various processes of software development differ based on how strictly the tasks are being addressed and on addressing its sequences.

In the late 1990s, the development of several agile methodologies arose, which means a move in the software development processes. Further, it indicates to the number of software development projects that are being currently organised [12]. In the 2000s, the significance of a software acting as discriminator for conventional products including products pertaining to the online software requires more rapid time-to-market times. Additionally, a significance of user-interactive products produces a rapid user feedback that is also significant, where formal processes are made extremely indeclinable. Accordingly, the development iterative processes acquired further considerations [13]. Nowadays, the majority of software processes are being created based on the use of existing software functionality and standard tools that are derived from open source libraries or commercial products. In general, nearly 30% of the software components are required to form a custom built, where more flexible processes are allowed to be implemented [14].

This study concentrates on the agile software development model. The agility literature, approaches, and trends are highlighted in the following sections. The structure of this research paper is organized as follows. Section 2 introduces the agility literature and highlights the importance of agile development over traditional methodologies. Section 3 analyses the well-known agile methods including XP, Scrum and other common agile methods while the research trends are presented in Section 4. Finally, Section 5 concludes the paper.

[1] See http://agilemanifesto.org/

2. AGILE LITERATURE

Earlier in the 2000s, an endless flow of alterations within the software industry was experienced. New technology has rapidly evolved where exchanged ideas spread among the developers of software within the global connectivity, which is derived by WWW. The technological potential is a result of intensive investments through the IT industry. Moreover, various different software applications are currently being enhanced for the consumer market, which require interfaces that are user-friendly. Thus, the user feedback must be promptly integrated with the improved process resulting with requirements that are changeable and unpredictable. In general, a rapid change is continuously being ingrained within software manufacturing [13]. Consequently, the ability to adapt with new requirements and speed-to-market is significant in order to efficiently perform through an uncertain environment [15]. Only shorter product life-cycles could definitely act with these challenges. The lightweight approaches emerged during the 1990s and introduced a reversed pole to the heavy-weight development approaches that are seen to be extremely rigid to efficiently improve a software for volatile project circumstances [16].

When the Agile Manifesto was published in 2001, the agile software development has acquired popularity. There exist various software vendors such as Adobe [17], SAP [18, 19], Microsoft [20], and many others that have implemented different agile methodologies over the past years ago. As a result, the agile software development appears nowadays as a mainstream development approach. This approach including a continuous attention on expert software developers within restricted validations pertaining to the development approach effectiveness. Several approaches and methods vary from non-agile and agile methods. Abrahamsson et al. [21] study the agile methods, which include cooperative (close communication with customers), incremental (small software releases), straightforward (the involved approaches are simple to understand, adjust and learn), and adaptive (the ability of producing changes within the last moment) software development method. Conboy [6] improves another often-cited definition according to an extensive agility investigation through other research domains. Based on the perspective of Conboy, agility consists of two concepts, which are leanness and flexibility. Agility not only integrates the ability of changing, but rather can motivate the ability of the project team to rapidly adapt to any particular change. Additionally, leanness is defined as the involvement of the apparent customer value based on quality, simplicity and economy.

Schmidt et al. [22] propose a different perspective for conceptualising the agility pertaining to the software development team based on its central development task organisation (e.g. implementation, software specification, software validation and design). Collaboration as well as iteration are suggested to represent the central behavioural markers that are related to the agile teams. The previous tasks are repeatedly iterated by the agile teams when many team members are being involved in the process. Additionally, the software is created, designed, implemented and validated by these teams into small steps including the whole team within the entire steps.

2.1. How agile development is different from traditional models?

The entire traditional process models (Waterfall, Spiral, RUP, etc) are in common based on their long iteration cycles, their large design upfront and specifications of the documents, and their heavyweight approach to process management [23]. Abrahamsson et al. [21] declare that any development approach is considered agile if it is incremental, straightforward, cooperative, adaptive. This is totally different from the conventional approaches used for software development. Nerur et al. [24] discussed the main differences between the conventional approaches and the agile development (see Table 1).

Table 1. *Conventional versus agile software development.*

	Conventional Models	Agile Development
Basic assumptions	Developed Systems are built through extensive and meticulous planning, and are fully predictable and specifiable.	Small teams develop high-quality software using the principles of continuous design improvement and testing, and based on rapid feedback and change.
Management style	Control and command	Collaboration and leadership
Control	Process focal	People focal

Knowledge management	Explicit	Implicit
Communication	Formal	Informal
Role assignment	Individuals	Self-organizing teams
Development model	Life-cycle models (Spiral, Waterfall, or with some variation)	Evolutionary-delivery models
Customer's role	Important	Critical

The agile software development's value in comparison with the conventional methods focuses on the interactions of the users and developers as one primary successful driver [25]. Although the Standish Group's reports [11] can be argumentative, they mention that the developed software using the agile approach has three times the success rate of the conventional Waterfall, and a much lower time and cost percentages. Based on the promises held by the agile practices, these methods provide the potential for enabling the teams of the software development to adapt to the ever changing requirements of the customer within high collaborative and interactive levels that could result with better outcomes pertaining the project [25].

2.2. Agile Manifesto

In 2001, a group of 17 advocates who are relying on lightweight software engineering methods, and who are grouped together in order to build the agile Manifesto [3, 26]. The Agile Manifesto produces a group of four core values that are appropriate for organisations that adopt the agility in software development. In the early 2000s, these core values were brought by previous lightweight methods that are provided by these agilists [3]. Therefore, the essence of agile development is formed by four values, which are comprised as:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

The concentration of the first value is on interactions and individuals, which implies that the agile software development team is considered as 'flexible and organic' instead of 'formalised, bureaucratic, and mechanistic" [24]. Decision-making can be delayed by specialized roles within the operational level based on different teams [27]. Here, the developers have self-organize, blend and interchange various roles [28]. As indicated by the second value, the documentation is deprioritized by the agile software development for the product in order to spend less on time for documentation in aiding a fast software delivery [26]. The concentration of the third value is based on how to successfully collaborate with the customer [28]. The decision-maker, as role for the project manager, is highly reduced [24]. The fourth value of change adaption means that the incremental and iterative features of the agile software development with various product releases enable the project teams to adopt and give prompt responses [29]. The agile manifesto concentrates on the customer collaboration and working software [30]. The manifesto aims at achieving the customer's satisfaction by performing a fast delivery for the product. The agile manifesto also concentrates on the delivery of the valuable software [31].

The focus of this study is to analyze the current agile methodologies to answer the following questions: is there any lack in the theoretical basics of the agile development? Are there significant benefits of using agile development over other conventional methods? and what is the best Agile method in use?

3. AGILE METHODS

Miller [32] mentioned the following set of features to the agile software processes based on fast delivery, which shortens the development life-cycles of projects:

1. Short cycles with Iterative that enable rapid corrections and verifications.
2. Modularity for the development process level
3. Iteration cycles ranging from 1 to 6 weeks.
4. Adaptive with current potential emergent risks.
5. An incremental process method allows creating and functioning applications through small steps.
6. The stinginess in the improved process eliminates the entire unneeded activities.
7. People-oriented, that is, the agile processes help users through any technologies and processes.
8. Incremental (and convergent) method attempts at reducing any risks.
9. Communicative and collaborative working style.

Various software engineering approaches were produced in the early 2000s. Such approaches rely on the ideas of an evolutionary, iterative and incremental software development processes [3]. Since these approaches include the four indicated core parameters of the agile software engineering, they were afterwards called agile methodologies. In this section, the recent state of the agility for software development methodologies are studied. Not only are Scrum and Extreme Programming the most influential, but are currently also considered the most common methods [3, 33]. There are many other approaches, which are either rediscovered or invented to refer to the same family of software development methodologies. Such approaches comprise Crystal Methods [34], Feature-Driven Development (FDD) [35], Adaptive Software Development (ASD) [16], Agile Unified Process (AUP) [36] and Lean Software Development (LSD) [37]. Fig. 1 shows the state-of-the-art agile methodologies, and in the following subsections, an overview of these agile methodologies is highlighted.

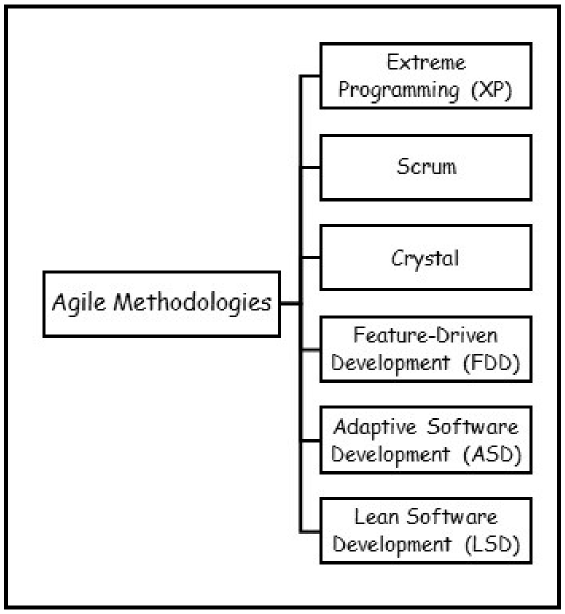
Agile Methodologies
Extreme Programming (XP)
Scrum
Crystal
Feature-Driven Development (FDD)
Adaptive Software Development (ASD)
Lean Software Development (LSD)

Fig. 1. *Agile methodologies*

3.1. Extreme Programming (XP)

The emergence of the XP has been commonly recognised as a start point for many different agile development approaches [21]. XP is mainly studied by the authors as a lightweight approach for small to medium-sized teams developing projects based on requirements that are rapidly-changing or vague [3]. Beck [38] creates a group of programming practices. The main ideas are focused on a set of practices, principles and values in which the developers have to use to develop the software responsiveness to change and its quality. The XP aims at delivering useful concepts and ideas pertaining to the software engineering to "extreme" levels degrees [38].

The XP has developed from problems that were caused based on the long development cycles of conventional development methodologies [39]. It begins with practices that are seen to be operative in processes that are related to software development [21]. The XP methodology is "theorised" according to the key practices and principles that are being used [38]. Despite the fact that the XP individual practices are not up-to-date, they are still lined up and collected in order to function together based on novel methods, and thus constituting a new development method.

The XP attracted an essential attention due to its importance in testing, simplicity, communication and its maintainable developer-oriented practices including its motivating name [40]. The XP programmers support a robust concentration on a software coding process rather than a documentation or plans. Additionally, the software quality is considered to be the basic concentration where the quality must be enduringly checked based on automated tests. Furthermore, XP programmers maintain simple design and avoid characteristics that are overmanned [3].

3.1.1. The XP process

XP applies an object-oriented approach since it has an effective improved model and involves a group of practices and rules that arise through the context of four framework activities [3]. These activities are planning, designing, coding and testing (Fig. 2). The planning activity starts with *listening*, which is a requirements-collecting activity that provides the ability of the technical members of XP team to understand the context of the business for the software and to obtain a broad feel for the needed outputs and the major functionality and features. The XP design follows a thorough principle, which is called the "Keep It Simple" (KIS) principle [3]. A simple design is frequently favoured over a complex representation. Additionally, the simple design provides an implemented guidance for a story that is the same as it is written, nothing more, nothing less. The XP method encourages *refactoring* — a development practice that is based on restructuring of the software implementation that could improve the software quality, i.e. its structure or readability, without having the software functionality changed. The aim is to increase the software long-term maintainability and extensibility.

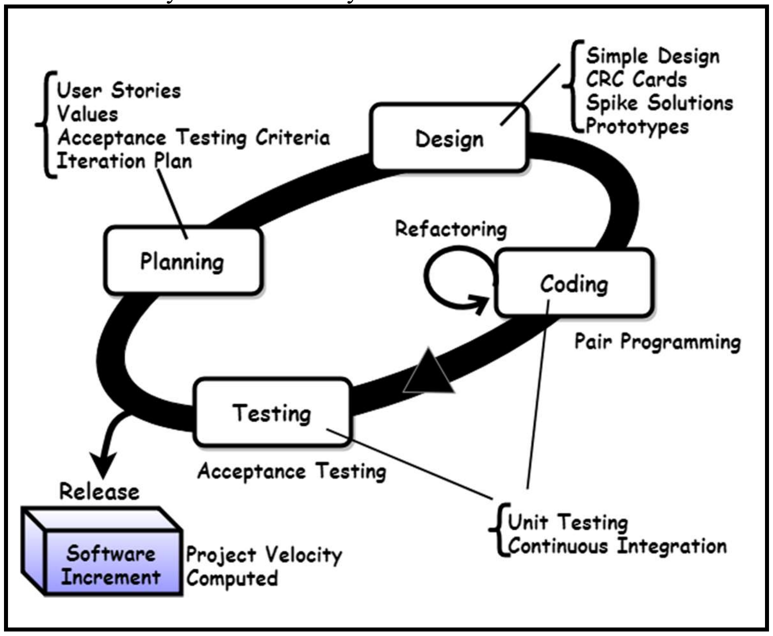

Fig. 2. *XP method*

The development team does not make a move to the code after performing the preliminary design work, but it instead creates a sequence of unit tests to practice every story that is involved within the new release or increment. When each unit test is developed, the development team can concentrate on what must be applied to efficiently pass the test. When completing the code, it can be directly unit-tested. The main idea of coding (and one of the most XP aspects) is to implement the so-called *pair programming*. It is recommended by the XP that a development team of two members share one computer and implement a side-by-side software. One developer writes the code, while the other developer challenges, supports and observes the selected method to obtain better results [3].

The unit tests are created before the start of the code, which is one of the key elements related to the XP [3]. The created unit tests must be applied in a way that allows them to be automated. This supports a strategy of a regression testing when modifying the code. Validation and integration testing that are applied for the system can be performed daily. The XP development team is provided with a continuous indication progress and can earlier increase the alerts if things go askew. The customer identifies the *acceptance tests* of XP*,* aka *customer tests,* to concentrate on the entire functionality and characteristics of the system, which are reviewable and visible by the customer. The *Industrial XP* (IXP) is variant of the XP and was proposed recently [41]. The XP is refined by the IXP and the agile process is targeted by the IXP when it is mainly being used through large organisations.

3.2. SCRUM

Scrum (taken its name from the rugby match) is considered to be an agile development approach, which is laid by J. Sutherland and his group during the 1990s [41]. Scrum is a project management framework [3] and relies on agile framework values and principles [2]. It is the most common agile-inspired development method that is frequently being applied [41]. Scrum (1) determines particular *roles* within the development team and (2) creates an iterative work mode, which is centralised through the development *sprints*, and (3) defines various *artefacts* for which are being used by the developers in order to organise their given tasks. The key elements of Scrum are shown in Fig. 3.

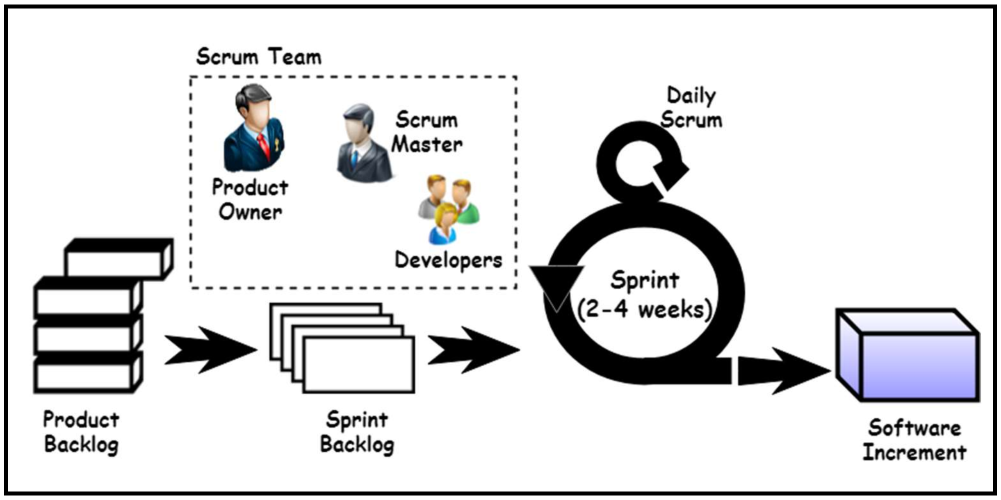

Fig. 3. *Scrum method*

The *Scrum team* consists of nearly ten persons (typically from six to 10 developers [1]). There are two particular roles relating the Scrum team, which comprise a *product owner* (PO) and a *scrum master* (SM) [3]. The *PO* represents the customer and voices his/her requirements. The PO role is considered to be a managerial role [2]. The PO defines the development targets in the coming sprint and is responsible of creating a value for customers [3]. The PO normally defines customer's requirements and a sequence of prioritised development tasks by ensuring the highest value items to be always on top [23]. The work increments are reviewed by the PO after each sprint. The SM acts as a facilitator who is in charge of maintaining scrum processes, and who could remove impediments that might stop the team from working in an efficient manner. The SM does not involve the responsibilities of people management, but rather it can behave as a teacher and a coach where it stresses that the scrum process is being followed [23].

The remaining members of the Scrum team refer back to the development team. These team members analyse the software requirements, design, implement and test the developed software [3]. The Scrum development team is considered to be cross-functional, i.e. all members are seen to include an essential skills set in order to perform the entire tasks pertaining to the software development. Subsequently, there is no extra role within the Scrum team such as testers or developers for the user interface. The development team members' size ranges from 2 – 7 persons [23].

An iterative work mode is followed by the Scrum teams where the development project is divided into small development iterations. These small development iterations are called the development *sprints* and contain a particular length duration ranging from one to four weeks, after which new software characteristics are delivered to the customer [3]. Each sprint begins with a *sprint planning* meeting and the Scrum members decide on the to-be-implemented software characteristics. Accordingly, different sub-tasks are determined and assigned by the team members for the individual developers. The entire team members set up a daily meeting for an approximate duration of 15 minutes (often called daily stand-up [23]) so that their work process could be synchronised and could gain transparency through the team members [3]. The entire developers tell the Scrum team members about the achievements they performed. These developers define the current work and take issues that are likely to be addressed by the team into account. Hence, every team member provides answers for three key questions whilst the meeting is being held. These questions comprise: (1) 'What had I accomplished yesterday?', (2) 'What will I do before the next Scrum meeting?' and (3) 'Are there any obstacles?' [23]. All sprints terminate with *sprint review* meetings when the progress of Scrum members is presented to the PO or is immediately provided to the customer. Further, a retrospective meeting is organised by the SM for the team such that possible improvements are discussed regarding to the future teamwork processes.

The development tasks are classified and organised by the team members based on the use of a *product backlog*. A product backlog consists of a sequence of prioritized tasks that is identified by the PO [3]. Items are provided to the product backlog at any time (this shows the way of how modifications are presented). The backlog is assessed by the product manager and priorities are updated as required [41]. This backlog is broken by the development team members into a set of *sprint backlog items* where the backlog progress is tracked by a *Burndown chart* when every sprint occurs [3]. The Burndown chart presents the ratio of the committed versus achieved backlog items.

A product (or software) increment, which is a shippable product state, is considered to be the work sum that is performed in the present sprint and in the other previous sprints. A definition of done indicates the case when a backlog item is completed [23]. Such a definition involves the minimum requirements that are based on the functionality or documentation tests relating to the developed increment. Due to the distinctive features of Scrum; such as the existence of daily stand-up and the review of the work increments after each sprint, the interest in Scrum is being increased throughout the time [30].

3.3. Other agile process methods

As previously indicated, the most commonly used agile process methods comprise the SCRUM and XP. Nonetheless, several other agile process methods are proposed where they are being used through the industry. The most common are Adaptive Software Development (ASD), Crystal methodologies, Feature Drive Development (FDD), Agile Unified Process (AUP), and Lean Software Development (LSD) [41]. In the following subsections, a very brief overview of these agile methods is highlighted.

3.3.1. Adaptive software development (ASD)

Jim Highsmith [42] suggested the Adaptive Software Development (ASD) as a technique that is by means created to build complex systems and software. The philosophical underpinnings of the ASD concentrates on the team self-organization and the human collaboration. Highsmith defines the ASD as a "life cycle" that includes three phases which comprise learning, collaboration and speculation. During the *Speculation* cycle, the project starts, and the *Adaptive-cycle planning* is performed. The Adaptive-cycle planning makes use of the project initiation information, which forms the mission statement of the customer, the constraints of the project (e.g. user descriptions or delivery dates), and basic requirements, in order to identify the sequence of released software increments.

Motivated people use *collaboration* in such a way their creative and talented output are increased. This approach is considered to be a recurring theme for every agile method [41]. When the ASD members start developing the components related to the adaptive cycle, they emphasize on "learning" and on progress toward a completed cycle. It is argued by Highsmith that software developers frequently overrate their particular understanding pertaining to the process, project, and technology where learning will help them in developing their real understanding levels.

3.3.2. Crystal

The term "crystal" is taken from the features relating to the geological crystals along with their own hardness, shape and colour. Cockburn [34] and Highsmith [43] introduced the crystal family of agile approaches to perform a software development method, which delivers a premium characteristic to "maneuverability." The Cockburn's characteristic refers to as "a resource-limited, cooperative game of communication and invention. The primary goal here is to deliver a useful, working software, where the secondary goal is setting up for the next game" [41].

In order to attain manoeuvrability, a set of methodologies are defined by Cockburn and Highsmith where each methodology contains core elements that are based on work products, process patterns, practice, and roles, which are distinct to each other [41]. A Crystal family is actually a group of agile processes, which are proved to act effectively through many different project types. The aim here is to permit the agile team members to choose the member that belongs to the crystal family and is the most appropriate for their environment and project.

3.3.3. Feature driven development (FDD)

Similar to the other agile methodologies, the FDD brings a philosophy in which (1) collaboration is emphasised among members within a team. (2) the project and problem complexities are managed based on the use of a feature-based decomposition and follow it by integrating the software increments, (3) text-based, graphical and verbal means are used by a communication of technical detail [41]. FDD emphasizes the activities of software quality assurance by assuring a strategy of using design and code inspections, the incremental development, the implementation of audits of software quality assurance, the use of different metrics, and the use of different patterns for analysing, designing and constructing the software. Different types of presentations and articles of the FDD are found at: www.featuredrivendevelopment.com.

3.3.4. Agile unified process (AUP)

The AUP adopts both the "iterative in a small" and the "serial in a large" philosophy in order to create the software [36]. When adopting the traditional Unified Process activities, which are *inception, elaboration, construction,* and *transition*, a serial overlay is provided by the AUP, where the serial overlay is the sequence of activities of software development, which provides the ability for a team to visualise the entire flow of the process for the software project. Nonetheless, based on every activity, the team iterates to deliver significant software increments for end users and to achieve agility as fast as possible.

3.3.5. Lean software development (LSD)

Lean production brings importance on value based on the reduction of costs, through removing "waste", where waste can be represented as large inventories and waiting time [30]. The LSD adapts the lean manufacturing principles to developing the software. The LSD principles that motivate the LSD work is briefly summarised as create knowledge, build quality in, deliver fast, eliminate waste, defer commitment, respect people, and optimize the whole [41]. Every principle is adapted to the process of the software. For instance, the "*eliminate waste*" principle is based on the context of the agile project. This could be interpreted as: (1) The addition of important functions or features, (2) The evaluation of the schedule and cost impact of any requirement that is currently requested, (3) The elimination of any extra processing steps, (4) The creation of mechanisms for developing a way in which team members can search for the information, (5) The ensuring that the testing process will find as many errors as possible, (6) The reduction of the time that is needed for requesting and obtaining a decision, which puts an impact on the software or on the process that creates it, and (7) The streamlining of the manner in a way that can transmit an information to the entire stakeholders who are engaged in the process.

4. TRENDS

It is clear that no single method is able to perform tasks for all projects [21, 44]. However, the project manager(s) must determine the nature of the project, and after that, the best appropriate development methodology is selected [21]. According to McCauley [45], both process-oriented and agile methods are significantly required since there is no one-size-fits-all development paradigm, which can be appropriate with the whole conceivable purposes. This view is common through many specialists in the field [46]. The principal aspects of agile and light methods comprise speed and simplicity. Hence, the development group just focuses on functions that are required within the development work, rapidly delivering them, gathering feedbacks and responding to the information that is being delivered. What lets a development methodology be an agile one? The case is based on a software development that is **cooperative** (developers and customers who are continuously working all together with close communications), **incremental** (small software releases with short iterations), **adaptive** (the ability of the method to produce last moment changes), and **straightforward** (the methodology is well documented, simple to learn, and easy to be modified).

Many studies provide productivity comparisons between the agile and conventional software development demonstrating positive results to an unbiased effect, while most of the researches show a positive effect on the quality [3, 47]. Critical researches on developing software using agility investigate the novelty related to using agility in software development, criticises a lack of concentration on a long-term architecture, claims that it is just appropriate to a small development team, and envisages that XO could yield with an ineffective teamwork [48]. Until now, the research community remains apart from completely comprehending how, why or in which perspectives of a project the agile software development performs [3]. Researchers carry out extensive studies that aim at improving a complete theoretical understanding of using the agility in the software development. The theoretical perception does not only clarify the success of the agility in the software development, but moreover, it leads professionals on the way of using the agile development method.

In summary, the agile software development is a development method that is rising with its popularity since the start of the 2000s. A study on the agile software development is improved from its best-practice achievement stories to stricter researches related to a character that is mainly descriptive. The availability of the knowledge that belongs to the teamwork research provides promising theoretical lens pertaining to the direction of the study. Since the high relevancy of the software development organisations with each other, such a method must not just be generalizable, but could further be validated with the data that is derived from expert software developers. While the study on agile software development is considered extremely fragmented, the dependent variable and the conceptualisation of agility are required to be clearly studied by each research study. Follow-up researches are based on results obtained from the study to improve the field towards a prospected and integrated research. Despite the robust concentration on the collaboration and teamwork within the agile teams, only few researches about the effectiveness of work teams is investigated in order to better understanding the agile methodologies.

Overall:
1. There is an urge motivation for stricter theory-supported researches along with different visions derived from expert software developers.
2. By implementing the theory over agile practices, it is possible to understand the agile activities value as methods that could rise the cooperation through the development team and through customers and developers [25].
3. The agile development introduces significant benefits. However, these benefits are not compatible with all projects, people, situations, and products [41].
4. Agility can be implemented through any software process. In order to achieve that, it is important to design the process in a way to give the teamwork the ability of streamlining tasks and adapting them together. Additionally, it performs a plan in such a way that comprehends the fluidity of the agile method. It removes the significant products and maintains them lean. Therefore, it ensures the incremental strategy of delivery that rapidly provides a workable software to customers as simple as it could be for the operational environment and the type of product.
5. For practitioners, it is noticed that there is a deterioration in the interest in extreme programming, while the interest in Scrum is being increased throughout the time [30].

5. CONCLUSION

Today, agile software development methods are considered lightweight methods that could employ an incremental and iterative lifecycle accompanied with short requirements and iterations, which could be modified within the development with broad participation by the customer. Many agile methods are proposed and developed, with the XP and Scrum considered as the two most commonly used agile methods. Every agile method consists of its own set of specified practices including many different concentrations. The XP, for instance, is comprised of practices that concentrates on different activities pertaining to the software development teamwork, whereas the Scrum possess a set of practices that improves the project management by rapidly revealing risks throughout the project. Trends for testing software development methodologies demonstrate that the practices of agility are adapted to the workplace context as organisations that adopts more practices of the agile-like software development. This study found that the agility practices are frequently underestimated because of the lack of the theoretical basics. The study results also reveal that the agile development introduces significant benefits over conventional methodologies. However, these benefits are not compatible with all projects, people, situations, and products. In addition, due to the distinctive features of Scrum, such as the existence of daily stand-up and the review of the work increments after each sprint, the interest in Scrum is being increased throughout the time, while the interest in XP is deteriorating. One of the options for the future research is to test the current agility practices that most commonly being used and to compare between these practices. This leads to an open question: "how do the agile practices provide values to the software development teams?"

REFERENCES:

[1] Patanakul, P. and R. Rufo-McCarron, *Transitioning to agile software development: Lessons learned from a government-contracted program.* The Journal of High Technology Management Research, 2018. **29**(2): p. 181-192.
[2] Steinhardt, G., *The Product Manager's Toolkit: Methodologies, Processes and Tasks in High-Tech Product Management*. 2010: Springer Science & Business Media.
[3] Schmidt, C., *Agile Software Development*, in *Agile Software Development Teams*. 2016, Springer. p. 7-35.

[4] West, D., et al., *Agile development: Mainstream adoption has changed agility.* Forrester Research, 2010. **2**(1): p. 41.

[5] Hoda, R., et al., *Systematic literature reviews in agile software development: A tertiary study.* Information and software technology, 2017. **85**: p. 60-70.

[6] Conboy, K., *Agility from first principles: Reconstructing the concept of agility in information systems development.* Information Systems Research, 2009. **20**(3): p. 329-354.

[7] Dingsøyr, T., et al., *A decade of agile methodologies: Towards explaining agile software development*. 2012, Elsevier.

[8] Curcio, K., et al., *Usability in Agile Software Development: A Tertiary Study.* Computer Standards & Interfaces, 2019.

[9] Sommerville, I., *Software Engineering. International computer science series.* ed: Addison Wesley, 2004.

[10] Radatz, J., A. Geraci, and F. Katki, *IEEE standard glossary of software engineering terminology.* IEEE Std, 1990. **610121990**(121990): p. 3.

[11] Clancy, T., *The Standish Group CHAOS Report.* Project Smart, 2014.

[12] Boehm, B. *A view of 20th and 21st century software engineering*. in *Proceedings of the 28th international conference on Software engineering*. 2006. ACM.

[13] MacCormack, A., R. Verganti, and M. Iansiti, *Developing products on "Internet time": The anatomy of a flexible development process.* Management science, 2001. **47**(1): p. 133-150.

[14] Royce, W., K. Bittner, and M. Perrow, *The economics of iterative software development: Steering toward better business results*. 2009: Pearson Education.

[15] Baskerville, R., et al., *Is internet-speed software development different?* IEEE software, 2003(6): p. 70-77.

[16] Highsmith, J. and A. Cockburn, *Agile software development: The business of innovation.* Computer, 2001. **34**(9): p. 120-127.

[17] Green, P. *Measuring the impact of scrum on product development at adobe systems*. in *System Sciences (HICSS), 2011 44th Hawaii International Conference on*. 2011. IEEE.

[18] Schnitter, J. and O. Mackert. *Large-scale agile software development at SAP AG*. in *International Conference on Evaluation of Novel Approaches to Software Engineering*. 2010. Springer.

[19] Schmidt, C.T., S. Ganesha Venkatesha, and J. Heymann. *Empirical insights into the perceived benefits of agile software engineering practices: A case study from SAP*. in *Companion Proceedings of the 36th International Conference on Software Engineering*. 2014. ACM.

[20] Begel, A. and N. Nagappan. *Usage and perceptions of agile software development in an industrial context: An exploratory study*. in *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*. 2007. IEEE.

[21] Abrahamsson, P., et al., *Agile software development methods: Review and analysis.* arXiv preprint arXiv:1709.08439, 2017.

[22] Schmidt, C., et al., *Team adaptability in agile information systems development.* 2013.

[23] Scheerer, A., *Coordination in Large-Scale Agile Software Development*. 2017: Springer.

[24] Nerur, S., R. Mahapatra, and G. Mangalaraj, *Challenges of migrating to agile methodologies.* Communications of the ACM, 2005. **48**(5): p. 72-78.

[25] Yu, X. and S. Petter, *Understanding agile software development practices using shared mental models theory.* Information and Software Technology, 2014. **56**(8): p. 911-921.

[26] Fowler, M. and J. Highsmith, *The agile manifesto.* Software Development, 2001. **9**(8): p. 28-35.

[27] Moe, N.B. and A. Aurum. *Understanding decision-making in agile software development: a case-study*. in *Software Engineering and Advanced Applications, 2008. SEAA'08. 34th Euromicro Conference*. 2008. IEEE.

[28] Drury-Grogan, M.L., K. Conboy, and T. Acton, *Examining decision characteristics & challenges for agile software development.* Journal of Systems and Software, 2017. **131**: p. 248-265.

[29] Dybå, T. and T. Dingsøyr, *Empirical studies of agile software development: A systematic review.* Information and software technology, 2008. **50**(9-10): p. 833-859.

[30] Dingsøyr, T. and C. Lassenius, *Emerging themes in agile software development:*

Introduction to the special section on continuous value delivery. Information and Software Technology, 2016. **77**: p. 56-60.
[31] Alahyari, H., R.B. Svensson, and T. Gorschek, *A study of value in agile software development organizations*. Journal of Systems and Software, 2017. **125**: p. 271-288.
[32] Miller, G.G. *The characteristics of agile software processes*. in *tools*. 2001. IEEE.
[33] Holvitie, J., et al., *Technical debt and agile software development practices and processes: An industry practitioner survey*. Information and Software Technology, 2018. **96**: p. 141-160.
[34] Cockburn, A., *Crystal Clear*. A Human-Powered Methodology for Small Teams, 2005.
[35] Felsing, J.M. and S.R. Palmer, *A Practical Guide to Feature-Driven Development*. IEEE Software, 2002. 7: p. 67-72.
[36] Ambler, S., *The agile unified process (aup)*. Ambysoft, 2005. 14.
[37] Poppendieck, M. and T. Poppendieck, *Implementing lean software development: From concept to cash*. 2007: Pearson Education.
[38] Beck, K. and E. Gamma, *Extreme programming explained: embrace change*. 2000: addison-wesley professional.
[39] Beck, K., *Embracing change with extreme programming*. Computer, 1999. 32(10): p. 70-77.
[40] Larman, C. and V.R. Basili, *Iterative and incremental developments. a brief history*. Computer, 2003. 36(6): p. 47-56.
[41] Pressman, R.S., and Maxim, Bruce, R., *Software Engineering: A Practitioner's Approach*. 2015, McGraw-Hill.
[42] Highsmith, J.R., *Adaptive software development: a collaborative approach to managing complex systems*. 2013: Addison-Wesley.
[43] Highsmith, J.A. and J. Highsmith, *Agile software development ecosystems*. Vol. 13. 2002: Addison-Wesley Professional.
[44] Hawrysh, S. and J. Ruprecht, *Light methodologies: It's Like Déjà Vu All Over Again*. Cutter IT Journal, 2000. 13(11): p. 4-12.
[45] McCauley, R., *Agile development methods poised to upset status quo*. ACM SIGCSE Bulletin, 2001. 33(4): p. 14-15.
[46] Glass, R.L., *Agile versus traditional: Make love, not war!* Cutter IT Journal, 2001. 14(12): p. 12-18.
[47] Wellington, C.A., T. Briggs, and C.D. Girard. *Comparison of student experiences with plan-driven and agile methodologies*. in *Frontiers in Education, 2005. FIE'05. Proceedings 35th Annual Conference*. 2005. IEEE.
[48] McAvoy, J. and T. Butler, *The role of project management in ineffective decision making within Agile software development projects*. European Journal of Information Systems, 2009. **18**(4): p. 372-383.