Journal of Computer and Communications, 2017, 5, 74-97
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Agile Software Development Methodologies: Survey of Surveys

Malek Al-Zewairi[1], Mariam Biltawi[1], Wael Etaiwi[1], Adnan Shaout[2]

[1]Computer Science Department, King Hussein Faculty of Computing Sciences, Princess Sumaya University for Technology (PSUT), Amman, Jordan
[2]The ECE Department, The University of Michigan-Dearborn, Dearborn, US
Email: malek.alzewairi@gmail.com, maryam@psut.edu.jo, w.etaiwi@psut.edu.jo, shaout@umich.edu

How to cite this paper: Al-Zewairi1, M., Biltawi, M., Etaiwi, W. and Shaout, A. (2017) Agile Software Development Methodologies: Survey of Surveys. *Journal of Computer and Communications*, **5**, 74-97.
https://doi.org/10.4236/jcc.2017.55007

Received: March 6, 2017
Accepted: March 28, 2017
Published: March 31, 2017

Abstract

Agile software design and development methodologies have been gaining rigorous attention in the software engineering research community since their early introduction in the mid-nineties in addition to being highly adopted by the software development industry. In the last 15 years, an excessive number of research studies have been conducted on agile methods, a great number of notable methods have been proposed and various surveys have been presented by many researchers. In this study, the authors intend to conduct a literature survey study of the surveys of the different agile methodologies ranging from January 2000 to December 2015 using an intuitive research methodology called "Compare and Review" (CR). Furthermore, these survey papers were classified into four major categories according to their area of study. Additionally, the newly proposed agile methodologies that have not been addressed yet in any other literature review were reviewed and compared in terms of where the changes that they proposed lay on the SDLC.

Keywords

Agile, Software Methods, Survey, Compare and Review, Research Methodology

1. Introduction

Agile in essence is an iterative, lightweight and lean software design and development methodology that was born in the late 1990s to be highly compatible with the rapid development of the WWW (World Wide Web) [1]. Similar to climbing a well-designed ladder where length of all steps and distance between each step is equivalent, agile methods divides a task into small-length iterations that have the same interval size and distance making the transition between iterations much smoother with much higher pace. Agile methodologies try to find

an equilibrium point between no process and too much process, allowing it to survive in dynamic environments where requirements frequently change while striving high quality software product [2]. Unlike other methods, agile methods rely on feedback as control mechanism which ensures greater customer satisfaction [3].

Agile encompasses various methodologies, including: Adaptive Software Development (ASD) [4], Agile Unified Process (AUP) [5], Crystal Methods [6], Dynamic Systems Development Methodology (DSDM) [7], eXtreme Programming (XP) [8], Feature Driven Development (FDD) [9], Kanban [10], Lean Software Development [11], Scrum [12], Scrumban [Ladas 2009 and several variant methods of agile].

The agile methodology is based on the "iterative enhancement" [13] technique [14]. As a iteration based methodology, each iteration in the agile methodology represents a small scale and self-contained Software Development Life Cycle (SDLC) by itself [1]. Unlike the Spiral model [15], agile methods assume simplicity in all practices [14].

In this research, the authors identify the following contributions: 1) a new research methodology called Compare and Review (CR) is used in this paper; 2) A survey of the surveys on agile methodologies were conducted, in which the survey papers were classified into four categories: "Agile Requirements Engineering", "Agile Methods", "Hybrid Agile Methods" and "Miscellaneous"; 3) Several new agile methods that have not been surveyed yet were reviewed and compared in terms of the changes that they proposed on the SDLC. Moreover, 26 surveys and 4 articles were selected for evaluation out of more than 92 studies. The results prove agile capability as a strong software design and development methodology.

The rest of the paper is organized as follows: Section 2 describes the research methodologies. Section 3 explores the surveys on agile methods in the literature and Section 4 reviews the most recent researches on XP, Scrum and FDD agile methodologies that, up-to-our-knowledge, have not been addressed in any other literature review. Finally, in Section 5, the evaluation of the survey studies and the new agile methods is presented, and then the conclusion is presented in Section 6.

2. Research Methodology

The authors followed a two-stage research methodology called "Compare and Review" (CR), where the first stage aims to compare the survey studies on agile methodologies. While, the second stage intends to review the most recent research studies on XP, Scrum and FDD agile methods that have not been addressed in any previous literature reviews. In the following subsections, the research methodology will be explained. **Figure 1** shows the flowchart for the CR research methodology.

2.1. Research Requirements

In this subsection, the research requirements that were used to govern this study

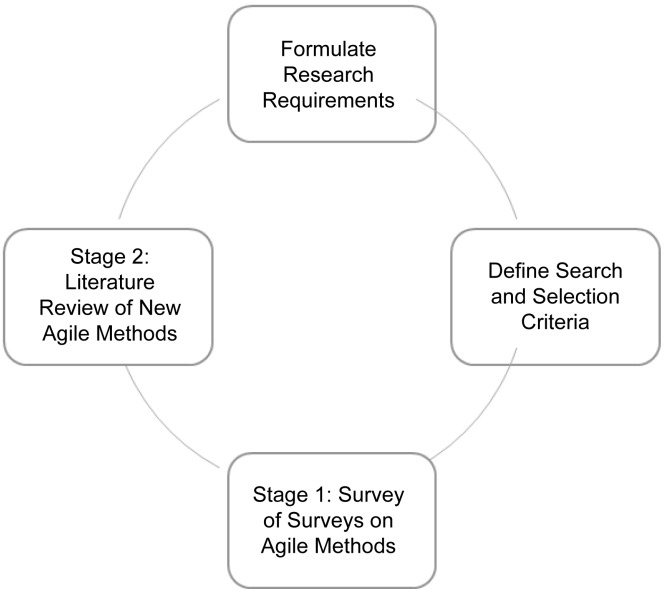

Figure 1. Flowchart of the CR research methodology.

are as follows:

- Formulate a basic understanding of the different Agile Software Development Methodologies.
- Formulate a comprehensive knowledge of the XP, Scrum and FDD agile methodologies.
- Formulate a comprehensive comparison between the newly proposed agile methods in terms of the changes that they might have made in SDLC phases.

2.2. Selection Criteria

In this subsection, the search and selection criteria that was applied is listed. The research must be:

- Indexed in one of the following digital databases: 1) ACM Digital Library; 2) IEEE Digital Library; 3) Science Direct; 4) Springer or 5) Wiley Online Library.
- Either a Conference Paper or Journal Article.
- Published between January 2000 and December 2015.
- In English language (the entire paper and not just the abstract).
- Has one of the following key words in the paper Title, Keywords list, or Abstract: {Agile; Software Methods; XP; Extreme Programming; Scrum; FDD; Feature-Driven Development; Literature Review; Survey}.

2.3. Stage 1: Survey of Surveys on Agile Methods

This stage aims to provide a comprehensive comparison between the various survey studies that addresses the different agile methodologies, which were published between January 2000 and December 2015. A strict search and selection criteria were applied, more than 32 studies were identified and 26 studies were selected for the final review.

2.4. Stage 2: Literature Review of New Agile Methods

In this stage, the authors reviewed 12 new research papers that have not been addressed yet in any previous literature review. Notably, all the papers were published in 2015 and the same selection criteria that was applied at stage 1 was applied in this stage too.

3. Survey of Surveys on Agile Methods

In this section, the authors will explore the different surveys that have addressed the agile methodologies in the past 15 years starting from January 2000 to December 2015. The surveys were classified into four categories as illustrated in Figure 2: "Agile Requirements Engineering", "Agile Methods", "Hybrid Agile Methods" and "Miscellaneous".

3.1. Agile Requirements Engineering

One the most attractive features of agile methodology is that it accepts changes to requirements during any phase of the SDLC, making it more flexible and highly adaptable to dynamic environments where requirements change frequently. In this subsection, the authors review the literature reviews on Agile Requirements Engineering (RE).

Baruah presented in [16] a comparative study about the different ways each of the agile methodologies manage requirements. Table 1 summarizes the differences between the eight agile software development methods in terms of requirements management as presented in [16].

Inayat, *et al.* presented in [17] a comprehensive systematic review about resolving traditional requirements engineering issues using agile requirements engineering for studies published between January 2002 and June 2013 in which 21 papers out of 531 were studied. The main results of the study can be summarized as follow:

- Distribution of the publication sources is 57% conferences, 19% journals, and 5% magazines.
- The studies can be classified into four main categories according to the specific topic that they address, which are: 1) agile RE practices (29%); 2) new methods (28%); 3) traditional RE vs. agile RE (5%), and 4) agile RE in general (38%).

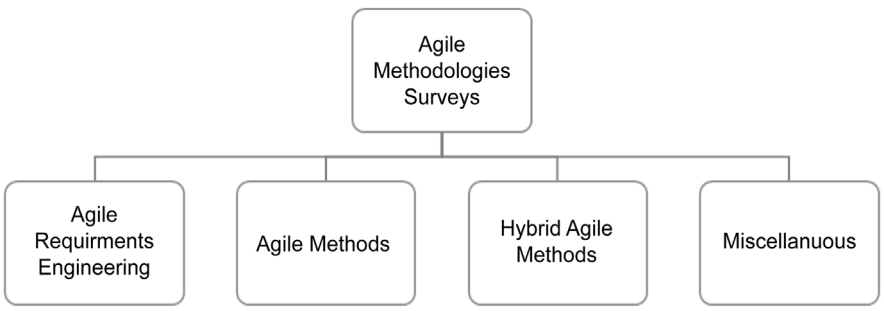

Figure 2. Agile methodologies surveys.

Table 1. Comparison between the different requirements management techniques under agile methods.

#	Agile Method	Requirements Representation	Customer Role
1.	XP	User Stories as both written cards and conversations. Written cards are not mandatory for implementation and are only considered "promises for conversation". Requirements are not supposed to be complete or clearly stated. User stories are destroyed after implementation is completed.	On-site customer is required to participate in requirements definition, estimation and prioritize.
2	Scrum	User stories are used to represent requirements. The actual requirements are defied based on the discussion of user stories between software owner and software developers.	Software owner plays the lead role in defining the requirements.
3	FDD	Requirements are represented using UML diagrams. List of features are used to manage the functional requirements. Requirements are first represented in a high-level context. For each modeling area, the requirements are modeled per domain. After requirements are modeled, it should be peer reviewed.	Not specified.
4.	Lean Software Development	Just-In-Time methodology is applied in requirements gathering. User stories (cards) are also used by the customer to specify initial requirements and sample screens by the developers. Developers then provide a time estimate for each card.	Provide input on sample screens and initial user stories.
5.	ASD	Requirements gathering is part of the speculation phase.	Not specified.
6.	Kanban	User stories are used to define each sprint main goal. Each sprint handles a single user story. Each story is divided into server-side and client-side task. Each task is further divided into subtasks.	Not specified.
7.	AUP	Requirement phase consists of the following activities: (1) identifying stakeholders, (2) understanding problem, (3) establishing a basis of estimation, and (4) defining user interface. User stories are used in the construction phase. Requirements are presented as Business Use Case Model.	Not specified.
8.	DSDM	Requirements are gathered and prioritized during the feasibility phase.	Not specified.

- Year 2011 has the most published papers (43%).
- The majority of studies are affiliated to authors from North America and EU.
- The majority of studies followed an exploratory research methodology.
- Seventeen common RE practices among the different agile methods were identified as shown in Table 2, which also indicates whether this practice is part of the original XP method or not.

Ramesh, *et al.* provided an empirical study on agile RE practices in [18]. The study-analyzed data collected from sixteen software companies (in USA) in order to explain the differences between traditional RE and agile RE. The data was solicited through systematic interviews. Moreover, a risk-based framework was proposed to evaluate the influence of utilizing agile RE through factoring the associated risks. The major findings of the study are presented as follow:

- Six agile RE practices were identified, which are: 1) Face-to-face communication instead of written specifications; 2) Iterative RE; 3) Using extreme practices in requirement prioritization; 4) Using constant planning to manage requirements changes; 5) Prototyping, and 6) Using review meetings with acceptance tests.

Table 2. Summary of common RE practices among Agile Methods.

#	Practices	Part of the Original XP
1.	Face-to-face communication	Open Workspace
2	Customer involvement	On-Site Customer
3	User stories	Planning Game
4.	Iterative requirements	Planning Game
5.	Requirements prioritization	Planning Game
6.	Change management	N/A
7.	Cross-functional teams	N/A
8.	Prototyping	N/A
9.	Testing before coding	Tests
10.	Requirements modelling	N/A
11.	Requirements management	N/A
12.	Review meetings and acceptance tests	Continuous Integration
13.	Code refactoring	Refactoring
14.	Shared conceptualizations	Metaphor
15.	Pairing for requirements analysis	N/A
16.	Retrospectives	Continuous Integration
17.	Continuous planning	Planning Game

- Seven problems posed by agile RE practices were identified, which are: 1) Difficulties with the estimation of cost and schedule; 2) Inefficient or unsuitable architecture; 3) Overlook of non-functional requirements; 4) Participation of customers; 5) Single dimension prioritization; 6) Insufficient requirements verification, and 7) Lacking sufficient documentation.
- Nine risks common with agile RE and its practices were identified by reviewing the literature. Which are: 1) Lacking requirements stability; 2) Problems with customer capacity and consent; 3) Insufficient interaction between customer and developer; 4) Missing important requirements; 5) Only modelling functional requirements; 6) Overlook requirements reviewing; 7) Using designs to present requirements; 8) Focusing on perfecting requirements prior commencing coding phase, and 9) Substantial flaws with the schedule.

3.2. Agile Methods

Stavru surveyed the usage of agile methods through industrial survey studies published between 2011 and 2012 [19]. They determined the papers, which could be trusted and recommend that the quality level of researches could be improved. The author concluded that the majority of the surveyed studies were incomprehensive in addition to being not trustworthy. On the other hand, the authors provided some recommendations in order to raise the research quality.

The recommendations include examining the rate of agile method usage as compared to alternative methods; examine the rate of agile method usage in an organizational level. Also, conducting researches about using agile method by academics beside industry in order to decrease the gap between industry and academia in addition to increase the trustworthiness in widespread adoption of agile method usage. Additionally, providing highly detailed reports in the future; thus, raising the level of confidence and trustworthiness in the reported studies.

Campanelli and Parreiras presented aspects of research on agile methods tailoring in [20]. The term agile method tailoring refers to the problem of selecting an agile method to be adopted in the organization. The authors analyze the total number of papers published in the area and proposed **Figure 3** that shows the number of articles published on agile method tailoring per year. The authors classified the selected studies into two main category groups: categories focusing on research aspects such as research type and research validation, and categories focusing on technical aspects such as agile method covered and criteria for method tailoring.

Dingsøyr, *et al.* summarized the prior research in agile software development and presented them within three main categories: the first category is agile principles and agility, which involves adapting agile by refining the development process to adapt to changes as needed. The second category is Research on agile software development, which involves agile software development processes. The last category is the seminal contributors and their relationships, which involves identifying relationships among sources of seminal information of agile software development [21].

da Silva Estácio and Prikladnicki studied the research studies that address Distributed Pair Programming (DPP) from educational or industrial perspectives in [22]. The term Pair Programming (PP) implies that two programmers are collaborating at one computer. The distributed word refers to the geograph-

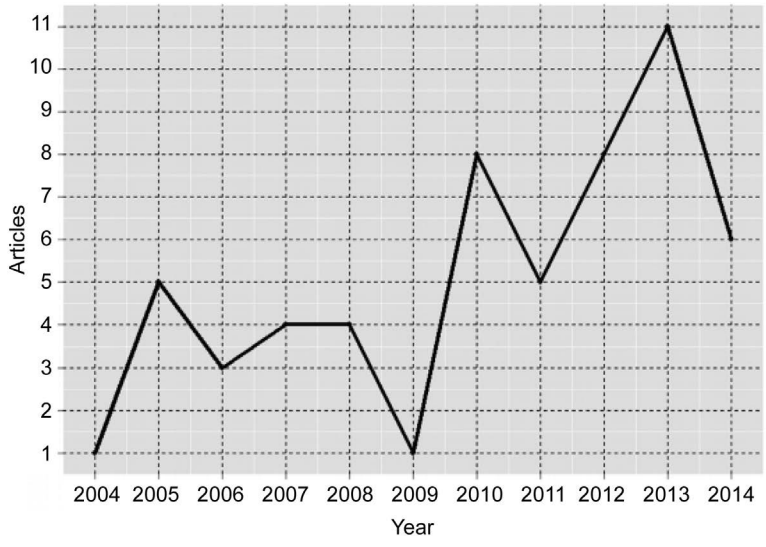

Figure 3. Number of papers on agile methods per year. Adopted from [20].

ical distribution of team members. The authors concluded that many quantitative and qualitative conclusions such as; Empirical studies involving DPP and real projects with industry professionals are very few and there are a need for more empirical studies. Few studies also investigate the effects of DPP with professionals, and the need to explore the effects of coordination, communication and cultural diversity in DPP.

Hamed and Abusham are viewed the most popular agile methods systematically and their appropriateness with regard to Small to Medium Enterprises (SMEs) environmental challenges in [23]. In this survey the authors proposed a definition and a discussion of Scrum, XP, Crystal family and Dynamic Systems Development Method (DSDM) methods, then explored the SME software challenges. The similarities and differences of these methods were compared against the defined criteria.

Salvador, *et al*. reviewed the studies related to usability techniques in agile methods, namely; scrum, XP and crystal in [24]. Moreover, 32 papers were surveyed, the results of the survey showed that most frequent techniques used with agile methodologies are the complementary techniques. The survey also concluded that most of the studies performed usability evaluations only during the implementation phase, which in turn were mostly constructed as elaborations of case studies.

Hummel conducted a systematic and structured literature review of agile Information Systems Development (ISD) in [25].Around 482 papers were collected and investigated, however the author extended the findings of previous three reviews by introducing new perspectives. The results illustrated a lack of agile ISD quantitative studies and theoretical underpinnings. While, XP is the most researched agile ISD method, scrum needs more research effort.

Dybå and Dingsøyr conducted a systematic review of 36 empirical studies of agile software development in the period from 2001 to 2005 to investigate the benefits, the limitations and the strength of evidence for agile methods. These studies were grouped into four main groups: introduction and adoption, human and social factors, perceptions on agile methods, and comparative studies. The focus was on the six agile methods; XP, Scrum, Crystal, DSDM, FDD, and Lean software development. The authors conclude that XP is the most common agile method investigated in the reviewed studies, and there is a need to investigate in the remaining agile methods. In addition to concern about the quality of studies provided [26].

Abrahamsson, *et al*. presented a comparative analysis of agile software development methods, including the method's life cycle coverage, project management support, practical guidance type, fitness-for-use, and empirical evidence as the analytical lenses in [27]. Moreover, the authors concluded that the majority of the studied methods did not concern about project management nor about improving the development team member's skills and capabilities. Finally, the authors recommend to concern about the quality of new methods instead of quantity of existing methods.

3.3. Hybrid Agile Methods

Selleri Silva, *et al*. presented a review study on the agile methodologies that integrate the Capability Maturity Model Integration (CMMI) where 3193 studies were identified and 81 were selected for evaluation and classified into two main classes; benefits to the organization in general and benefits to the development process. The results shows that using agile methods was helpful in reaching level 2 and level 3 of CMMI and in some cases even level 5 [28].

Torrecilla-Salinas *et al*. studied the applicability of complying with the CMMI-DEV model for Web development companies that have followed one of the agile methods [29]. The study surveyed the current state-of-the-art on this topic to answer five questions that were later used to analyze and evaluate the selected studies. Six papers were selected for evaluation out of more than 1453 studies. The results have shown that in the last 5 years more and more Web development companies are moving towards adopting agile methods in order to help be certified against CMMI-DEV.

Santana, *et al*. provided a literature review to identify software process improvement (SPI) in agile environment [30]. The authors classified the reviewed papers according to SPI aspects. Additionally, they identified new distinct approaches to Agile SPI, and they have suggested using one of the following three agile SPI approaches: top-down approach, the agile SPI based on improving behavior, and the agile SPI based on improving practices. The authors also identified the difference between traditional and agile SPI especially in their goals. The goal of traditional SPI is to elaborate a repeatable process that could be improved with lessons learned. On the other hand, the process in agile SPI should be ready for changes and improve its capability for changes. Furthermore, the author mentioned another difference that is related to the knowledge transfer policies; the traditional SPI dictates that knowledge should be transferred to people based on some organizational policies and criteria such as training and document use, while agile SPI considers knowledge transfer through meetings, informal and learning should be based on individual experiences of team members.

Literature on the application of different agile practices in Global Software Engineering (GSE) was summarized in [31]. The term GSE refers to distributing agile across cultural, temporal, and geographical boundaries. The authors classify researches according to research type, as shown in **Figure 4**. They conclude that the majority of studies are in the form of experience reports that contains the experience in particular issues. On the other hand, there is a need for providing more validation, and evaluation research. Another conclusion the authors mentioned that there is a need for analyzing the challenges and advantages of combining Agile and GSE in the form of evaluation research.

An overview of the use of software metrics in the industrial agile context is provided by [32]. This study makes three contributions: First, the authors categorize the metrics found in empirical agile studies and compares them with the metrics suggested by agile literature, and they conclude that agile teams use many metrics suggested by the agile literature. Second, the study highlights the reasons for and

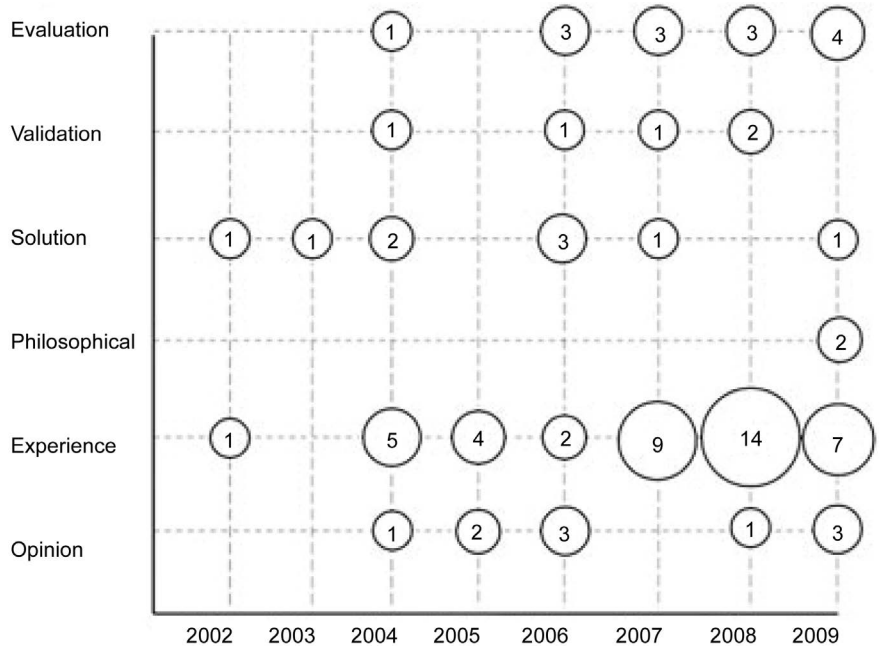

Figure 4. Distribution of research types over the studied years. Adopted from [31].

effects of using metrics in agile software development. They conclude that metrics are used in the following areas: Sprint and Project Planning, Sprint and Project Progress Tracking, Understanding and Improving Quality, Fixing Software Process Problems, and Motivating People. Third, the study identifies high influence metrics based on the number of occurrences and statements found in the primary studies, they conclude that Velocity, Effort estimate, and Defect count were the most popular metrics, and the most important metrics according to qualitative analysis of metric importance are Customer satisfaction, Technical debt, Build status, and Progress as working code.

Integrating agile software development processes with User Centered Design (UCD) was presented in [33] as a systematic literature review, in order to identify and classify various challenging factors that restrict Agile and User Centered Design Integration (AUCDI) such as: Lack of time for upfront activities, and to explore the proposed practices to deal with these challenges. The authors classify the finding in this topic according to publication channel and publication year.

Sriram and Mathew presented a review of literature on applying agile methodologies in Global Software Development (GSD) and how agile methodology fit in GSD [34]. Three main ideas were identified in this review paper on GSD; performance of global software development, governance related issues, and software engineering process related issues. More distant analysis of literature on both agile and GSD showed that various types of agile methods were applied and tailored appropriately to produce optimal performance in the context of GSD, while empirical studies addressing GSD-Agile fit were not found. The most common agile method used in GSD is SCRUM as the authors conclude.

Sohaib and Khan presented a literature review to describe how usability fit with agile software development in order to gain stronger and effective usable

software system [35]. The authors mentioned that the usability could be adapted to agile software development by doing some steps such as: using more iterations and concern about testing at each phase of SDLC.

Using agile methods in the embedded software projects is reviewed systematically by Shen, *et al*. in [36]. The authors concluded that there is a need for more effective research is this area. Although, there are many difficulties, Xie, *et al*. concluded that using agile in such projects holds a positive impression [37]. The authors presented many challenges of applying agile in embedded software projects, such as: development team challenges, time constraints and budget. The authors recommended choosing the appropriate agile method according to the challenges derived from the embedded system and how can the selected agile method tackle these challenges.

3.4. Miscellaneous

Sletholt, *et al*. conducted a literature review to investigate the effects of using agile practices in scientific software development processes, focusing on evaluating the agility of scientific software projects presented in five carefully selected papers. The authors defined and utilized an agile mapping chart, with elements based on Scrum and XP reference models for agility assessment purpose. The authors compare their findings with the previously provided surveys. The findings of these comparisons indicated that scientific software development projects that have adopted agile have an improved testing process to compare to the traditional methods [38].

A new analytical framework developed by Qumer and Henderson-Sellerscalled (4-DAT) [39]. The proposed framework was applied on six agile selected methods in addition to two traditional methods for comparison purposes. The evaluation approach is done in this analytical framework to evaluate these methods at the process level and practice level from four perspectives. The evaluation aimed to select an appropriate agile method for a particular development.

Rauf and Al Ghafees provided an industrial survey to study the use of scrum and XP agile practices in computer application development, the authors posted a questionnaire to investigate the benefits of using agile, 79 responses were collected from 45 companies. Survey results showed that 57% respondents use Scrum and in turn it is the most common agile method used, while 27% respondents use both XP and Scrum and only 5% respondents have reported to use XP solely. In addition, the authors investigated the strength of benefits against each agile practice, and explored the challenges against each agile practice [40].

A web-based survey conducted by Begel and Nagappan of employees who are working on the software production processes [41]. The survey investigated how Microsoft employees use agile software development methods and how they penetrate of agile software development practices and their perceptions of why agile works well or poorly on their software teams. The employee's responses indicate that around one-third of the respondents use agile, and SCRUM is the most popular method with 65% of the respondents were using it in their team,

and most of agile users have a positive opinion about it.

4. Literature Review

In the following subsections, the authors will briefly introduce three agile methods (*i.e.* XP, Scrum and FDD), then review the most recent researches that, up- to-our-knowledge, have not been addressed in any other literature review yet.

4.1. Extreme Programming (XP)

Among all of the agile methods, Extreme Programming (XP) is the most popular and well-documented method [14] [23]. The XP method was first proposed by Beck as a last resort to rescue a project that had been declared a failure [8]. In XP, simplicity is the driving factor that applies to the software development practices including the communications with the customer. Beck has defined twelve rules that govern the XP method [14] [42] that are:

- Planning Game: the planning starts at the beginning of each iteration where the stakeholders of the project meet to define, estimate and prioritize the "User Stories" (*i.e.* requirements) for the next release.
- Small Releases: there are two types of version releases: initial version and working version. The initial version is produced after a few iterations and it does not implement all features, but only essential ones. While, a working version is produced after a few weeks and contains most of the features.
- Metaphor: it is used in the modelling of the software system and is constructed by all the stakeholders.
- Simple Design: It is the base of the XP methodology and applies to the requirements gathering, system design, coding, and communications with the customer.
- Tests: In XP methodology, testing is considered one of the major activities to ensure high quality product in addition to high customer satisfaction. Testing begins before the coding phase, where developers are required to prepare the test functions prior writing the code itself. While, customers are required to prepare the functional test scenarios for each iteration.
- Refactoring: It means that any changes made on the system must uphold the simplicity feature.
- Pair Programming: Coding of the software system is carried out in a group of two developers.
- Continuous Integration: New parts of the software system are integrated as soon as they pass both the unit and functional test cases.
- Collective Ownership: The ownership of the code produced belongs to all the developers.
- On-Site Customer: Someone from the customer side must work with the development team at all times.
- 40-Hour Weeks: The maximum working hours per week for developers must not exceed 40-hours, which implies that the requirements must be revised to

adhere to this rule.

- Open Workspace: All work, including coding and development must take place in a common environment.

Following, the review of the most recent works on XP that, up-to-our-knowledge, have not been addressed in any other literature review yet.

Chen and Wu proposed a modified method of XP consisting of 11 steps called "my Agile" in order to allow more integration between XP methodology and other computer science topics such as data structure, Object-Oriented Programming (OOP), algorithms, etc. The 11 steps proposed method ("my Agile") shares 4 steps with the original XP method, which are step 5: "Dispatching and Scheduling", step 6: "Unit Test Code", step 9: "Coding" and step 10: "Unit Testing and Acceptance Testing". Although, the authors identify steps 0, 1, 2 and 3 ("Exploring Requirements", "Scenarios", "Acceptance Test Cases and User Manual", and "CRC Session" accordingly) as adopted from the Software Engineering discipline, steps 0 and 1 can be easily mapped to the first rule (*i.e.* planning the game) of the original XP method. While, step 3 can be partly mapped to the fifth rule (*i.e.* testing). Although, the remaining three steps 4, 7 and 8 ("Reverse Engineering Tool", "Data Structure Design", and "Algorithm Design" accordingly) are identified as novel steps designed specifically for "my Agile", step 4 can be seen as merely using Computer-Aided Software Engineering (CASE) tools to automate some of the design and verification steps. A project aiming to develop a student grading system was designed to evaluate the proposed methodology and it was applied in four universities in Taiwan [43].

Haryono presented a case study where XP methodology was applied in a project for developing a Financial Management System (FMS) as a part of the E-Government (e-Gov) in Indonesia [44]. The project followed the core practices of XP methodology as shown in **Figure 5**. The project was evaluated using a satisfaction questionnaire. The results showed that 100% of respondents were satisfied with the method and indicated that it has helped with communications. Moreover, it showed an increase in the sense of belonging to the system. Nonetheless, 2% of respondents declared it less fit to the project type.

Radhakrishnan, *et al*. proposed a generic software model for educational purposes that improves on the Common Software Measurement Integration Consortium—Full Function Point (COSMIC FFP) method [45]. The method is designed to measure the functional size of software, including real-time software to better estimate project resources and schedules. The proposed model "eXtreme Software Teaching" (XSOFT) integrates the XP method with the COSMIC FFP method to help bridging the effort and time gap between learning software development and working in software development.

4.2. Scrum

Scrum is a management and control process used for developing and sustaining complex products in order to build software that meets business needs, incrementally and empirically. It is considered a widely used agile method, first

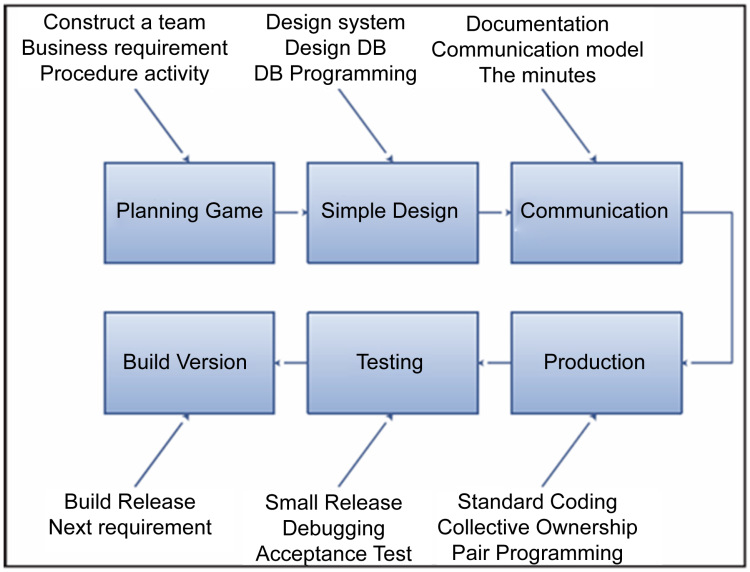

Figure 5. Development methodology of the Indonesian E-gov FMS project. Adopted from [44].

described in the year 1996 [12]. Scrum is also considered a lightweight, simple to understand and difficult to master method.

Scrum starts by splitting the project into iterations (sprints). Before each sprint, in the planning phase, all tasks to be done are kept in a list called "release backlog". During the planning process a next-sprint goal is identified and announced to the developers in order to show them the tasks are being performed and at which level of detail to implement them, in addition, a prioritized collection of tasks are selected from release backlog to be completed in the next sprint. When the planning phase is complete, each development team carries their tasks.

The tasks in the sprint backlog remain unchanged until the end of the sprint phase. While the development teams develop their tasks, the project should be tracked and monitored through daily meetings and track tasks status in order to enhance communications between teams and keep focus on the overall project goals. After every sprint, an analysis and evaluation process progress through pre-sprint meeting.

Following, the review of the most recent works on Scrum that, up-to-our-knowledge, have not been addressed in any other literature review yet.

Chandana Ranasinghe and Perera demonstrated how the challenges and issues related to offshore development (OSD) in Sri Lankan context can be overcome using the scrum method, and to achieve success in OSD it is important to combine engineering practices with scrum [46].

Esteves Maria, *et al.* described the use of the scrum agile method and its best practices in the development of several academic interdisciplinary projects, which are (1) a Java application prototype, based on Big Data, IoT, and (2) Credit Card fraud detection for a Proof of Concept (PoC), using cloud-compu- ting resources accordingly [47].

Scott, *et al*. aimed to support the meshing hypothesis through using teaching strategies matching the Felder-Silverman Learning Style Model in a Scrum course and focusing on the processing dimension of this model. Consequently, the authors corroborated that the knowledge of Scrum of the undergraduate students' was improved when they were given suitable instructional methods according to the processing dimension of their learning styles. The authors provided experiments as evidence to support the meshing hypothesis [48].

Pierre Mattei, *et al*. used a combination of scrum agile methods with model-based programming to overcome the restraints of developing Space on-board system project, which is considered complex, time consuming, and highly susceptible to errors [49].

de Souza, *et al*. presented an evaluation of Scrum adaptations to evaluate the capstone project [50]. A case study is also presented that illustrates the adoption of Scrum to manage the capstone project, which represented a direct and objective approach in order to have an environment similar to the real one. The authors' proposal was to determine, discuss and quantify how flexible and collaborative Scrum becomes when teaching Software Engineering.

Pauly, *et al*. presented a study to assess the adoption or adaption of Scrum principles at an e-commerce company. The authors presented an in-depth single case study, which in turn revealed that not all scrum principles are suitable in each context [51].

The relationship between SCRUM and SDLC practices was studied in [52]. The authors proposed a framework to utilize the user experience design in SDLC in organizations that use SCRUM in association with Capability Maturity Model Integration (CMMI) practices. Moreover, the authors utilized the user experience design dimensions recommended by the Human Factors Institute, which include training of professionals, creating and managing metrics to evaluate the usability, and establishing a successful cases database for training purposes. Many studies focus on using the scrum agile method in software companies, such as [53] that study using of SCRUM in Brazilian small business.

Raj, *et al*. proposed a modified scrum process that focuses more on testing, using Test-as-a-Service (TAAS) implementation in order to get the results faster without increasing the cost of the project [54].

4.3. Feature-Driven Development (FDD)

FDD methodology is one of the AGILE methods for software development. FDD is an iterative and incremental method based on dividing the software into many different features (models), and then builds each model separately. The development process for each model (feature) consists of five activities: develop general model, build feature list, plan for feature, design for feature and build by feature.

In the first activity, develop overall model, general high-level overview of the project is set to better understand the problem domain, this step is im-

portant to the team in order to manage the relationship and interaction between the team members and the customer. In build feature list activity, the team uses the knowledge extracted from the first activity to build a list of features (or functionalities) required, and categorize them according to business subjects.

The next activity is to plan for feature, in this stage, the development plan set by the project manager and development manager, the plan contains the ownership of feature, the time schedule and the responsibilities of the feature development team. In the next activity, design the feature, the focus moves to the features itself according to the programming tasks, the classes defined, the sequence diagrams provided and methods prologue defined. The final activity is to build by feature where each feature developed.

FDD is used for large projects, because it can be divided into many small-size tasks, which increase the possibility to complete the project successfully. This also gives the management the feasibility to change the team while the project running is without affecting the project time schedule and the overall quality. On the other hand, the documentation task is more complex, and the overhead of the chief programmer is high, because he acts as a coordinator, mentor and lead designer.

Mahdavi-Hezave and Ramsin proposed the Feature-Driven Methodology Development (FDMD) for Situational Method Engineering (SME), where the object-oriented features used to specify the requirement of the target methodology of SME in order to facilitate the development of the target methodology, and enhance maintainability and reusability [55].

ISMAIL, *et al.* compared the difference between using FDD and SCRUM methodologies to gather requirements for Open Source Software (OSS) projects [56]. The authors concluded that FDD team gets more time than SCRUM to accept changes in requirements, because FDD divided changes according to its severity while SCRUM accepted changes in requirement at any stage of the development process. The authors conclude that both the methodologies set the tasks priorities with different roles, in FDD the domain expert does this task, while in SCRUM it is done be scrum master. The authors compare the two methodologies according to customer interaction too. They conclude that the customer interact with teams at every phase in SCRUM, and in the initial phase in FDD.

5. Evaluation

In this section, the evaluation of the reviewed studies is categorized into two main categories; survey of surveys and new agile methods.

5.1. Evaluation of the Survey of Surveys

In this survey, the authors study several surveys related to agile software development. **Table 3** summarizes twenty-sex survey papers related to agile software

Table 3. Review of the different surveys using agile methods.

Ref.	Year	Type (C/J)	Publisher	Category	Year Range	# of Papers	Methods Covered
[27]	2003	C	IEEE	Agile Methods	1994-2002	N/A	Scrum, XP, Crystal, FDD
[26]	2008	J	Elsevier	Agile Methods	2001-2005	36 of 1996	Scrum, XP, Crystal, FDD, DSDM, Lean
[21]	2012	J	Elsevier	Agile Methods	2001-2011	N/A	N/A
[23]	2013	C	IEEE	Agile Methods	2001-2013	7 of 167	Scrum, XP, Crystal, DSDM
[19]	2014	J	Elsevier	Agile Methods	2011-2012	9	Scrum, XP, Agile Modeling, FDD, ASD, DSDM, TDD, Crystal Methods, Agile Up
[24]	2014	C	ACM	Agile Methods	2002-2012	32 of 307	Scrum, XP, Crystal
[25]	2014	C	IEEE	Agile Methods	2001-2013	482	Scrum, XP
[20]	2015	J	Elsevier	Agile Methods	2002-2014	56	Custom agile methods
[22]	2015	J	Elsevier	Agile Methods	2001-2014	34	DSDM
[18]	2010	J	Wiley	Agile RE	N/A	N/A	N/A
[16]	2015	J	Elsevier	Agile RE	1998-2014	13	XP, Scrum, FDD, Lean, ASD, Kanban, AUP, DSDM
[17]	2015	J	ACM	Agile RE	2002-2013	21	N/A
[35]	2010	C	IEEE	Hybrid Agile Methods	2002-2009	N/A	Scrum, XP
[31]	2011	J	Wiley	Hybrid Agile Methods	1999-2009	81	N/A
[36]	2012	C	IEEE	Hybrid Agile Methods	N/A	52	Scrum, XP, Crystal, FDD, DSDM, ASD, TDD
[34]	2012	C	IEEE	Hybrid Agile Methods	2000-2011	N/A	Scrum, XP, Crystal, DSDM, ASD
[37]	2012	C	ACM	Hybrid Agile Methods	N/A	72	Scrum, XP, Crystal, FDD, DSDM, ASD, TDD
[33]	2014	C	ACM	Hybrid Agile Methods	2000-2012	71	Scrum, XP
[28]	2015	J	Elsevier	Hybrid Agile Methods	1998-2011	81	XP, General agile, Scrum, Lean, Custom agile methods
[30]	2015	C	IEEE	Hybrid Agile Methods	2002-2012	31	Agile SPI and traditional SPI
[32]	2015	J	Elsevier	Hybrid Agile Methods	2002-2013	30	N/A
[29]	2016	J	Elsevier	Hybrid Agile Methods	2009-2014	6 of 1453	XP, Scrum, Lean, CMMI, Custom agile methods
[41]	2007	C	IEEE	Miscellaneous	N/A	N/A	Scrum, Agile Software Development
[39]	2008	J	Elsevier	Miscellaneous	1996-2006	N/A	Scrum, XP, Crystal, FDD ASD, DSDM
[38]	2011	C	ACM	Miscellaneous	2003-2009	5 of 573	Scrum, XP
[40]	2015	C	IEEE	Miscellaneous	N/A	N/A	Scrum, XP

development in the last 15 years. The table indicates the reference number for each paper, the paper published year, the paper type (whether the paper is published in either a conference (C) or a journal (J)). Additionally, Table 3 demonstrates the publisher and shows the survey category used in the article according to classification illustrated in Figure 2. Furthermore, it illustrates the period covered by the survey paper in addition to the number of articles surveyed in each reference. Finally, it lists the methods used in each survey.

As illustrated in Table 3, the number of surveys in agile software development increased in the last three years, the year 2015 alone has eight papers, six journal articles and two conference papers were published in reputable databases. The

reason of this increase refers to the increasing number of methods and enhancements of the Agile methods; thus, the need to compare, analyze and summarize the increasing amount of researches in Agile methods become a very important topic to the researcher and to the software development industry in order to improve their performance to gain better output quality. Figure 6 shows the number of survey papers published per year.

Interestingly, the distribution of paper type across conference papers and journal articles was divided equally as shown in Figure 7.

Figure 8 presents the ratio of published papers per publisher. It shows that Elsevier has the highest percentage of 38%, while Wiley has the lowest percentage of 8%.

Notably, "Hybrid Agile Methods" category was the most of surveyed in the literature showing the importance of agile method compared to the other development methodologies. Figure 9 shows the percentage of survey papers based on the four classification categories.

Lastly, XP and Scrum Agile methods were the most surveyed agile methods. Over 5529 papers were addressed in total in all of the 26 survey papers.

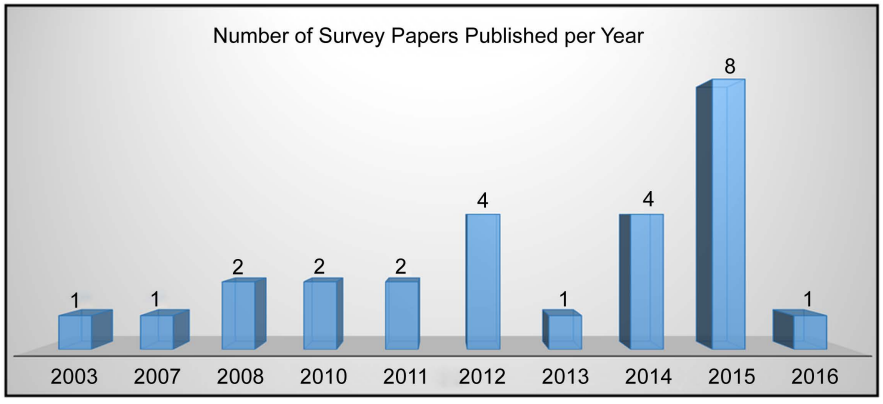

Figure 6. Number of survey papers published per year.

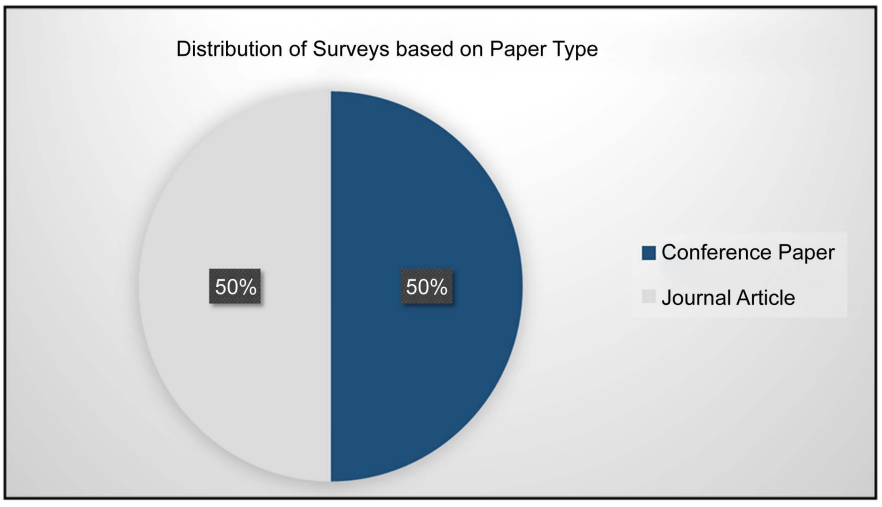

Figure 7. Distribution of surveys based on paper type.

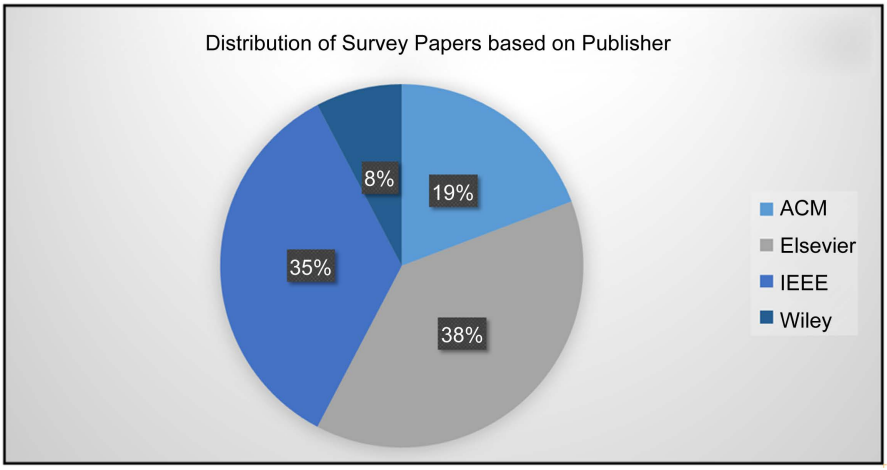

Figure 8. Distribution of survey papers based on publisher.

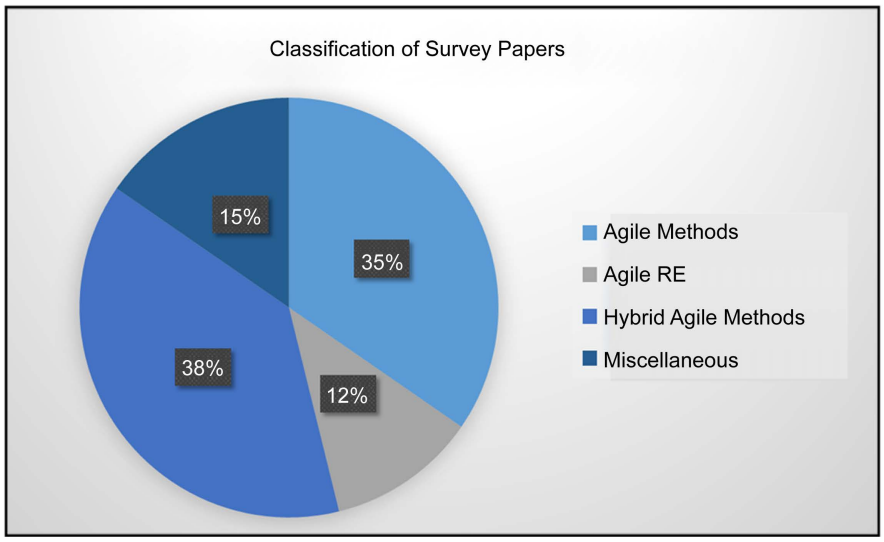

Figure 9. Percentage of survey papers based on their classification.

5.2. Evaluation for the New Agile Methods

Table 4 summarizes the evaluation for the new agile method reviewed in Section 5. It compares the new agile methods according to the changes made on the software development life cycle (SDLC). The results shows that all of the newly proposed methods ignore improving the coding stage, while, focusing on improving both the requirements and testing stages. As explained in Section 4(A), although the authors of "my Agile" claim to have improved on the original XP method, the authors were able to dispute their claim and map most of the changes back to the original XP. On the other hand, "XSOFT" did not proposed any changes on the original XP method. Instead, it proposed integrating the XP method with the COSMIC FFP method to in order to minimize the gap between learning software development and working in software development. As explained in Section 4(B), "Modified Scrum process" method proposed outsourcing the testing phase by utilizing the cloud testing services (TAAS) in order to improve the software testing results without increasing the cost of the project.

Table 4. Comparison between the different Agile Methods in terms of changes made tothe SDLC.

Ref.	Method Name	Agile Method	Req.	Design	Coding	Testing
[43]	"myAgile"	XP	Yes	Yes	No	Yes
[45]	"XSOFT"	XP	No	No	No	No
[54]	"Modified Scrum process"	Scrum	No	No	No	Yes
[55]	"FDMD"	FDD	Yes	No	No	No

As explained in Section 4(C), "FDMD" utilized common OOP practices in the requirements elicitation.

6. Conclusion

Agile is considered one of the most popular software design and development methodologies. In this study, a literature survey study of the surveys of the different agile methodologies ranging from January 2000 and December 2015 has been conducted. In this study, 26 survey studies were selected for review and evaluation using a new proposed research methodology called "Compare and Review". The surveyed studies classified into four categories: "Agile Requirements Engineering", "Agile Methods", "Hybrid Agile Methods" and "Miscellaneous". Moreover, four newly proposed agile methodologies were reviewed, analyzed and compared. The evaluation shows that most of surveys were proposed and published in 2015, and the most surveyed category were the Hybrid Agile methods.

References

[1] Williams, L. (2010) Agile Software Development Methodologies and Practices. *Advances in Computers*, **80**, 1-44. https://doi.org/10.1016/S0065-2458(10)80001-4

[2] El-Haik, B.S. and Shaout, A. (2010) Software Design for Six Sigma: A Roadmap for Excellence. Wiley, Hoboken.

[3] Highsmith, J. and Cockburn, A. (2001) Agile Software Development: The Business of Innovation. *Computer*, **34**, 120-127. https://doi.org/10.1109/2.947100

[4] Highsmith, J. (1997) Messy, Exciting, and Anxiety-Ridden: Adaptive Software Development. *American Programmer*, **10**, 23-29.

[5] Ambler, S. (2006) The Agile Unified Process (AUP) http://www.ambysoft.com/unifiedprocess/agileUP.html

[6] Cockburn, A. (2004) Crystal Clear a Human-Powered Methodology for Small Teams. Addison-Wesley, Reading.

[7] Tuffs, D., Stapleton, J., West, D. and Eason, Z. (1999) Inter-Operability of DSDM with the Rational Unified Process. *DSDM Consortium*, **1**, 1-29.

[8] Anderson, A., Beattie, R. and Beck, K. (1998) Chrysler Goes to Extremes. *Disruted Computers*, 24-28.

[9] Coad, P., de Luca, J. and Lefebvre, E. (1999) Java Modeling Color with Uml: Enterprise Components and Process with Cdrom. Prentice Hall, Upper Saddle River.

[10] Ladas, C. (2009) Scrumban-Essays on Kanban Systems for Lean Software Development. Modus Cooperandi Press, Seattle.

[11] Poppendieck, M. and Poppendieck, T. (2003) Lean Software Development: An Agile Toolkit. Addison-Wesley, Boston.

[12] Schwaber, K. (1996) Controlled Chaos: Living on the Edge.

[13] Basili, V.R. and Turner, A.J. (1975) Iterative Enhancement: A Practical Technique for Software Development. *IEEE Transactions on Software Engineering*, **SE-1**, 390-396. https://doi.org/10.1109/TSE.1975.6312870

[14] Cohen, D., Lindvall, M. and Costa, P. (2004) An Introduction to Agile Methods. *Advances in Computers*, **62**, 1-66. https://doi.org/10.1016/S0065-2458(03)62001-2

[15] Boehm, B. (1986) A Spiral Model of Software Development and Enhancement. *Software Engineering Notes*, **11**, 14-24. https://doi.org/10.1145/12944.12948

[16] Baruah, N. (2015) Requirement Management in Agile Software Environment. *Procedia Computer Science*, **62**, 81-83. https://doi.org/10.1016/j.procs.2015.08.414

[17] Inayat, I., Salim, S.S., Marczak, S., Daneva, M. and Shamshirband, S. (2015) A Systematic Literature Review on Agile Requirements Engineering Practices and Challenges. *Computers in Human Behavior*, **51**, 915-929. https://doi.org/10.1016/j.chb.2014.10.046

[18] Ramesh, B., Cao, L. and Baskerville, R. (2010) Agile Requirements Engineering Practices and Challenges: An Empirical Study. *Formation Systems Journal*, **20**, 449-480. https://doi.org/10.1111/j.1365-2575.2007.00259.x

[19] Stavru, S. (2014) A Critical Examination of Recent Industrial Surveys on Agile Method Usage. *Journal of Systems and Software*, **94**, 87-97. https://doi.org/10.1016/j.jss.2014.03.041

[20] Campanelli, A.S. and Parreiras, F.S. (2015) Agile Methods Tailoring—A Systematic Literature Review. *Journal of Systems and Software*, **110**, 85-100. https://doi.org/10.1016/j.jss.2015.08.035

[21] Dingsøyr, T., Nerur, S., Balijepally, V. and Moe, N.B. (2012) A Decade of Agile Methodologies: Towards Explaining Agile Software Development. *Journal of Systems and Software*, **85**, 1213-1221. https://doi.org/10.1016/j.jss.2012.02.033

[22] Da Silva Estácio, B.J. and Prikladnicki, R. (2015) Distributed Pair Programming: A Systematic Literature Review. *Formation and Software Technology*, **63**, 1-10. https://doi.org/10.1016/j.infsof.2015.02.011

[23] Hamed, A.M.M. and Abushama, H. (2013) Popular Agile Approaches in Software Development: Review and Analysis. 2013 *International Conference on Computing, Electrical and Electronics Engineering*, Khartoum, 26-28 August 2013, 160-166. https://doi.org/10.1109/ICCEEE.2013.6633925

[24] Salvador, C., Nakasone, A. and Pow-Sang, J.A. (2014) A Systematic Review of Usability Techniques in Agile Methodologies. *Proceedings of the 7th Euro American Conference on Telematics and Information Systems*, Valparaiso, 2-4 April 2014, 171-176.

[25] Hummel, M. (2014) State-of-the-Art: A Systematic Literature Review on Agile Information Systems Development. 2014 *47th Hawaii International Conference on System Sciences*, Waikoloa, 6-9 January 2014, 4712-4721. https://doi.org/10.1109/HICSS.2014.579

[26] Dybå, T. and Dingsøyr, T. (2008) Empirical Studies of Agile Software Development: A Systematic Review. *Formation and Software Technology*, **50**, 833-859. https://doi.org/10.1016/j.infsof.2008.01.006

[27] Abrahamsson, P., Warsta, J., Siponen, M.T. and Ronkainen, J. (2003) New Directions on Agile Methods: A Comparative Analysis. *25th International Conference on Software Engineering*, Portland, 3-10 May 2003, 244-254.

[28] Selleri Silva, F., Soares, F.S.F., Peres, A.L., de Azevedo, I.M., Vasconcelos, A.P.L.F., Kamei, F.K. and De Meira, S.R. (2015) Using CMMI Together with Agile Software Development: A Systematic Review. *Formation and Software Technology*, **58**, 20-43. https://doi.org/10.1016/j.infsof.2014.09.012

[29] Torrecilla-Salinas, C.J., Sedeño, J., Escalona, M.J. and Mejías, M. (2016) Agile, Web Engineering and Capability Maturity Model Integration: A Systematic Literature Review. *Formation and Software Technology*, **71**, 92-107. https://doi.org/10.1016/j.infsof.2015.11.002

[30] Santana, C., Queiroz, F., Vasconcelos, A. and Gusmao, C. (2015) Software Process Improvement in Agile Software Development: A Systematic Literature Review. 2015 41*st Euromicro Conference on Software Engineering and Advanced Applications*, Funchal, 26-28 August 2015, 325-332. https://doi.org/10.1109/SEAA.2015.82

[31] Jalali, S. and Wohlin, C. (2012) Global Software Engineering and Agile Practices: A Systematic Review. *Journal of Software-Evolution and Process*, **24**, 643-659. https://doi.org/10.1002/smr.561

[32] Kupiainen, E., Mäntylä, M.V. and Itkonen, J. (2015) Using Metrics in Agile and Lean Software Development—A Systematic Literature Review of Industrial Studies. *Formation and Software Technology*, **62**, 143-163. https://doi.org/10.1016/j.infsof.2015.02.005

[33] Salah, D., Paige, R.F. and Cairns, P. (2014) A Systematic Literature Review for Agile Development Processes and User Centred Design Integration. *Proceedings of the* 18*th International Conference on Evaluation and Assessment in Software Engineering*, London, 13-14 May 2014, 51. https://doi.org/10.1145/2601248.2601276

[34] Sriram, R. and Mathew, S.K. (2012) Global Software Development Using Agile Methodologies: A Review of Literature. 2012 *IEEE International Conference on Management of Innovation and Technology*, Bali, 11-13 June 2012, 389-393.

[35] Sohaib, O. and Khan, K. (2010) Integrating Usability Engineering and Agile Software Development: A Literature Review. 2010 *International Conference on Computer Design and Applications*, Qinhuangdao, 25-27 June 2010, V2-32-V2-38.

[36] Shen, M., Yang, W., Rong, G. and Shao, D. (2012) Applying Agile Methods to Embedded Software Development: A Systematic Review. 2012 2*nd International Workshop on Software Engineering for Embedded Systems*, Zurich, 9 June 2012, 30-36. https://doi.org/10.1109/SEES.2012.6225488

[37] Xie, M., Shen, M., Rong, G. and Shao, D. (2012) Empirical Studies of Embedded Software Development Using Agile Methods: A Systematic Review. *Proceedings of the* 2*nd International Workshop on Evidential Assessment of Software Technologies*, Lund, 19-20 September 2012, 21-26. https://doi.org/10.1145/2372233.2372240

[38] Sletholt, M.T., Hannay, J., Pfahl, D., Benestad, H.C. and Langtangen, H.P. (2011) A Literature Review of Agile Practices and Their Effects in Scientific Software Development. *Proceedings of the* 4*th International Workshop on Software Engineering for Computational Science and Engineering*, Wuhan, 9-11 December 2011, 1-9.

[39] Qumer, A. and Henderson-Sellers, B. (2008) An Evaluation of the Degree of Agility in Six Agile Methods and Its Applicability for Method Engineering. *Formation and Software Technology*, **50**, 280-295. https://doi.org/10.1016/j.infsof.2007.02.002

[40] Rauf, A. and AlGhafees, M. (2015) Gap Analysis between State of Practice and State of Art Practices in Agile Software Development. *Agile Conference*, Washington DC, 3-7 August 2015, 102-106. https://doi.org/10.1109/agile.2015.21

[41] Begel, A. and Nagappan, N. (2007) Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. 1*st International Symposium on Empirical Software Engineering and Measurement*, Madrid, 20-21 Sep-

tember 2007, 255-264. https://doi.org/10.1109/ESEM.2007.12

[42] Beck, K. (1999) Extreme Programming Explained: Embrace Change. Addison-Wesley, Boston.

[43] Chen, J.J.-Y. and Wu, M.M.-Z. (2015) Integrating Extreme Programming with Software Engineering Education. 2015 38*th International Convention on Information and Communication Technology, Electronics and Microelectronics*, Opatija, 25-29 May 2015, 577-582. https://doi.org/10.1109/MIPRO.2015.7160338

[44] Haryono, K. (2015) The Extreme Programming Approach for Financial Management System on Local Government. 2015 *International Conference on Science and Technology*, Pathum Thani, 4-6 November 2015, 29-34. https://doi.org/10.1109/TICST.2015.7369335

[45] Radhakrishnan, P., Kanmani, S. and Nandhini, M. (2015) XSOFT: A Generic Software Teaching and Learning Model. *Computer Applications in Engineering Education*, **23**, 432-442. https://doi.org/10.1002/cae.21613

[46] Chandana Ranasinghe, R.K. and Perera, I. (2015) Effectiveness of Scrum for Offshore Software Development in Sri Lanka. *Moratuwa Engineering Research Conference*, Moratuwa, 7-8 April 2015, 306-311.

[47] Esteves Maria, R., Rodrigues, L.A., Guarino de Vasconcelos, L.E., Fonseca Mancilha Pinto, A., Takachi Tsoucamoto, P., Nunweiler Angelim Silva, H., Lastori, A., Da Cunha, A.M. and Vieira Dias, L.A. (2015) Applying Scrum in an Interdisciplinary Project Using Big Data, Internet of Things, and Credit Cards. 2015 12*th International Conference on Information Technology—New Generations*, Las Vegas, 13-15 April 2015, 67-72. https://doi.org/10.1109/ITNG.2015.17

[48] Scott, E., Rodríguez, G., Soria, Á. and Campo, M. (2016) Towards Better Scrum Learning Using Learning Styles. *Journal of Systems and Software*, **111**, 242-253. https://doi.org/10.1016/j.jss.2015.10.022

[49] Pierre Mattei, A.L., Marques da Cunha, A., Vieira Dias, L.A., Fonseca, E., Saotome, O., Takachi, P., Sousa Goncalves, G., Pivetta, T.A., da Silva Montalvao, V., Kendi, C., Lopes de Freitas, F., Alves Ferreira, M. andrade Almeida, M. and Goncalves de Oliveira Rodrigues, G. (2015) Nanosatellite Event Simulator Development Using Scrum Agile Method and Safety-Critical Application Development Environment. 2015 12*th International Conference on Information Technology—New Generations*, Las Vegas, 13-15 April 2015, 101-106. https://doi.org/10.1109/itng.2015.22

[50] De Souza, R.T., Zorzo, S.D. and da Silva, D.A. (2015) Evaluating Capstone Project through Flexible and Collaborative Use of Scrum Framework. *IEEE Frontiers in Education Conference*, El Paso, 21-24 October 2015, 1-7. https://doi.org/10.1109/fie.2015.7344249

[51] Pauly, D., Michalik, B. and Basten, D. (2015) Do Daily Scrums Have to Take Place Each Day? A Case Study of Customized Scrum Principles at an E-Commerce Company. 2015 48*th Hawaii International Conference on System Sciences*, Kauai, 5-8 January 2015, 5074-5083. https://doi.org/10.1109/HICSS.2015.601

[52] Lima Peres, A. and Lemos Meira, S. (2015) Towards a Framework That Promotes Integration between the UX Design and SCRUM, Aligned to CMMI. 2015 10*th Iberian Conference on Information Systems and Technologies*, Aveiro, 17-20 June 2015, 1-4. https://doi.org/10.1109/cisti.2015.7170443

[53] Lisi Romano, B. and Delgado Da Silva, A. (2015) Project Management Using the Scrum Agile Method: A Case Study within a Small Enterprise. 2015 12*th International Conference on Information Technology—New Generations*, Las Vegas, 13-15 April 2015, 774-776. https://doi.org/10.1109/ITNG.2015.139

[54] Raj, G., Yadav, K. and Jaiswal, A. (2015) Emphasis on Testing Assimilation Using

Cloud Computing for Improvised Agile SCRUM Framework. 2015 *International Conference on Futuristic Trends on Computational Analysis and Knowledge Management*, New Delhi, 25-27 February 2015, 219-225. https://doi.org/10.1109/ablaze.2015.7154995

[55] Mahdavi-Hezave, R. and Ramsin, R. (2015) FDMD: Feature-Driven Methodology Development. 2015 *International Conference on Evaluation of Novel Approaches to Software Engineering*, Barcelona, 29-30 April 2015, 229-237. https://doi.org/10.5220/0005384202290237

[56] Ismail, U., Qadri, S. and Fahad, M. (2015) Requirement Elicitation for Open Source Software By Using SCRUM and Feature Driven Development. *International Journal of Natural & Engineering Sciences*, **9**, 38-43.