Strengths and Weakness of Traditional and Agile Processes — A Systematic Review

Mahrukh Sameen Mirza, Soma Datta*
University of Houston-Clear Lake, Houston, Texas, USA.

* Corresponding author. Tel.: 12182833838 email: datta@uhcl.edu, mahrukh.smirza@gmail.com

Manuscript submitted February 25, 2019; accepted April 10, 2019.
doi: 10.17706/jsw.14.5.209-219

Abstract: In the software industry, there are several processes and methodologies that exist. The traditional processes and Agile methodologies have their own strengths and weaknesses. Agile methodologies overcome some of the weaknesses of traditional processes. Although in the recent years Agile methodologies have been used by software development companies, there is still a high ratio of software failures when compared with core engineering processes. The adoption of these processes in software development could alleviate software failures. This systematic study reviews the strengths and weaknesses of both traditional processes and Agile processes. The search strategy resulted in 91 papers, of which 25 primary studies are investigated between 2012 and 2019. The detailed search strategy has been presented in this study along with future directions.

Key words: Agile, core engineering processes, extreme programming, feature driven development, Kanban, lean, scrum, systematic review, test driven development, traditional.

1. Introduction

Before 2001[1]-[13], the software industry used traditional software development processes (i.e., Classical waterfall model, iterative waterfall model, spiral model, RAD model). While these traditional models are known to be cost saving for bigger, off-shore projects, there is criticism that exists [13]-[25]. Due to these criticisms and the high ratio of software failures that used traditional models, it led to a change in software process development in 1999. This evolution started to encourage lightweight processes and a broader approach for better software development.

In 2001 [1], the original contributors of this evolution met and tried to identify the areas that these existing software methodologies had in common. Focusing on this common ground, led to the "Agile Manifesto".

Since the introduction of the "Agile Manifesto" in 2001, Agile methodologies have gained much popularity and success. The software industry had a huge shift from practicing traditional software development to now adopting Agile methodologies. There are several reasons why the software industry has chosen Agile over traditional models. Some of the reasons are faster product delivery, iterations, customer satisfaction, high product quality, etc. In general, the shift in Agile methodologies focuses more on individuals and interactions over processes and tools, working process over detailed documentation, customer collaboration over contact negotiation and responding to change rather than following a plan [20], [22]. It was also seen that Agile software development could handle changing requirements flexibly [15]. This method basically put more focus on quality product development, simplicity and enhancing knowledge from change incorporation. Several Agile methodologies such as Scrum, eXtreme programing (XP), and Lean work well with the organization but also have potential risks associated with them. These Agile methodologies are often easily misunderstood. It is also difficult to manage methods like Scrum in an organization because it requires all team players to be motivated.

The core engineering processes are well defined and followed properly because they are usually life critical products. Consider an example of the civil engineering discipline, it is important that the engineers properly design and deliver whatever they make. People think software development is different from the engineering design practice. While every other discipline follows almost the same engineering process, software development has different approaches towards development. If these engineering design approaches are integrated in the present software development practices, then there would be less failure and the process would turn out to be more advantageous.

The rest of the paper is organized as follows – section 2 consists of the research process that was used to do this systematic review. It provides keywords that were used to search for research papers and the inclusion and exclusion criteria used to select important papers. Section 3 consists of the research question. Section 4 has the related work and section 5 is the conclusion and future directions.

2. Research Process

There were 91 papers found by using the search keywords as in Table 1. Of these 91 papers, 72 papers were selected by reading the abstract. 25 papers were selected for primary study by using the inclusion and exclusion criteria. The search process which involved identifying the relevant papers was done by using different combinations of keywords. The search took place in steps in which the very first step was to search papers using the search keywords. The quality of the papers was determined after reading the papers in detail. Both the authors searched and downloaded the papers. Author 2 was responsible for reading the abstract and deciding if a particular paper was relevant or not by using the inclusion and exclusion criteria. Author 1 was responsible for determining the quality of the papers by reading them in detail. A summary of each paper after detailed reading was created by author 1.

The search keywords used to find the papers are show in Table 1 and Fig. 1 shows the phases that were involved in the searching process.

Table 1. Search Keywords

Subject	Search keywords
Traditional processes	Traditional software development OR traditional Agile OR software development life cycle OR SDLC OR traditional models OR traditional model OR traditional software model OR traditional software models OR waterfall Agile
Agile methodologies	Agile methodologies OR Agile software OR Agile development OR XP Agile OR eXtreme programming Agile OR Scrum Agile OR Crsytal Agile OR DSDM Agile OR dynamic system development method Agile OR FDD Agile OR feature driven development Agile OR Lean Agile OR Kanban Agile OR Agile manifesto
Engineering	Core engineering design process

The inclusion criteria was as follows –

1) Papers were published between 2012 to 2019
2) Papers written in English
3) Papers that were scholarly & peer reviewed and journal articles
4) Papers having computer science and engineering discipline

5) Papers having search terms software engineering, software and engineering
6) Papers where the search terms were found in the abstract
7) Papers that spoke about Agile, traditional or core engineering design process

The exclusion criteria was –

1) Papers that are duplicates of papers that were already included
2) Papers that did not talk about traditional, Agile or core engineering design process
3) Papers that were older than 2012

Table 2 below shows the papers that were included in the study. The table provides a brief description of each paper along with the year they belong to.

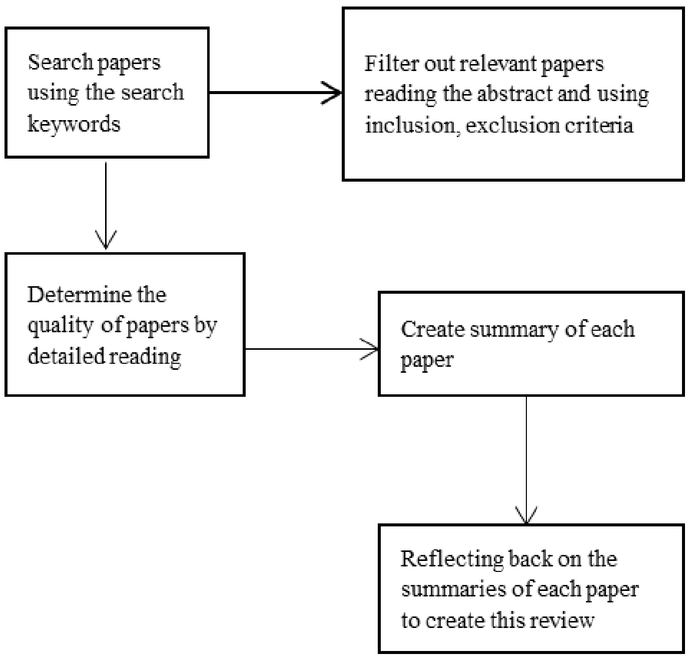
Search papers using the search keywords
Filter out relevant papers reading the abstract and using inclusion, exclusion criteria
Determine the quality of papers by detailed reading
Create summary of each paper
Reflecting back on the summaries of each paper to create this review

Fig. 1. Systematic search process.

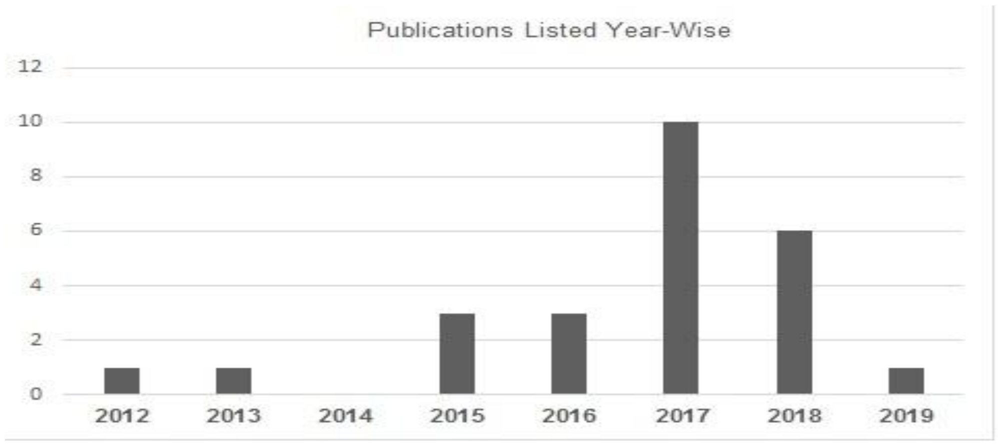

Fig. 2. Publication by year.

The fig above shows that there was one paper from 2019, six papers from 2018, ten papers from 2017, three papers from 2016, three from 2015, one from 2013, and one from 2012 and there were none from 2014.

Table 2. Brief Description of each Paper with Year

Year	Title	Brief description
2019	Lean and Agile software process improvement in traditional and Agile environment	The main objective of this paper is to show that both Lean and Agile approaches can be used depending upon what type of environment we are working on. The paper provides an overview of these approaches and identifies the well-known practices of both.
2018	Kanban in Software Engineering : A systematic mapping study	This paper conducts a systematic mapping of Kanban in software engineering between the years 2006 to 2016. From experience reports and primary studies, both the benefits and challenges of Kanban are identified.
	The Proposed L-Scrumban Methodology to Improve the Efficiency of Agile Software Development	In this paper, a new methodology of L-Scrumban is proposed which integrates Scrum, Lean and Kanban. The paper also validates this methodology which confirms its efficiency.
	Do Pair Programming Approaches Transcend Coding? Measuring Agile Attitudes in Diverse Information Systems Courses	In this paper, a tailored programming challenge is applied to a group of first year students through senior Information Systems (IS) and non – IS majors. This is to analyze how the attitudes of participants and perceived benefits of programming language change. It also determines whether the quality and functionality of the solutions differ across educational levels and disciplines.
	Back to the future: origins and directions of the "Agile Manifesto" – views of the originators	A survey and an interview study with the original contributors of Agile manifesto are presented in this paper. The paper talks about today's perspective and the outlook on future of the manifesto.
	What Do We (Really) Know About Test – Driven Development	This paper talks about Test Driven Development. It answers questions like Is TDD better than any other development method? Does it really fulfill all promises it makes? How do you decide whether or not to use TDD? And what are TDDs secondary effects?
	On the benefits and challenges of using Kanban in software engineering: a structured synthesis study	The goal of this paper was to present the benefits and challenges of Kanban for the practitioners so that they can understand and analyze them for real-time projects.
2017	SXP: Simplified Extreme Programming Process Model	In this paper, a simplified version of XP is presented which promises to overcome its drawbacks.
	Agile Software Development Models TDD, FDD, DSDM, and Crystal Methods: A Survey	This paper gives a review about TDD, FDD, Crystal and DSDM. It talks about phases that are involved in these processes, about misconceptions, advantages and disadvantages of each.
	Can FOSS projects benefit from integrating Kanban: a case study	There is a lack of research in integration of Free and Open Source Software (FOSS) and Agile Software Development (ASD). This paper attempts to integrate both and reports its benefits.
	Applicability and issues in traditional model of ERP implementations: an industry perspective	This paper presents a literature review of the ERP model and the real-time scenarios of practitioners when they work with this model.
	A Study of Software Development Life Cycle Process Models	A complete explanation of the SDLC models is given in this paper. The paper talks about Waterfall model, Iterative Model, Spiral Model, V-Model, Big Bang Model, Agile Model, Rapid Application Development Model and Software Prototype. The advantages and disadvantages are stated along with situations in which the model will best fit.
	Investigating Agile Adoption for Project Development	The main goal of this paper is to highlight Agile transition in companies along with project management challenges. It also presents a comparison of traditional and Agile software development

Year	Title	Brief description
	Simplified FDD Process Model	The main problems in FDD are that it is less responsiveness towards changing requirements, inappropriate for small scale projects and reliant on experienced staff. To overcome these limitations, this paper proposes a Simplified FDD Process Model.
	Assisting the continuous improvement of Scrum projects using metrics and Bayesian networks	In order to provide a quality assessment of Scrum projects, a process has been presented in this paper followed by Bayesian network. This process can be used by the Scrum masters for the improvement of business value delivery.
	Evaluating the Quality of Proposed Agile XScrum Model	XP and Scrum processes are integrated in this paper in order to enrich the strengths of both and overcome their limitations. The paper also validates the XScrum process by performing three case studies of industrial projects.
	Managing the requirements flow from strategy to release in large scale Agile adoption: a case study at Ericsson	An in-depth study of the Ericson telecommunications node development organization is presented in this paper. The study describes how the requirements flow beginning with strategy to release. Its related benefits and problems are also highlighted in the paper.
2016	Perceived barriers to effective knowledge sharing in Agile software teams	Based on an in-depth multi case study, this paper investigates how the project manager, developers, testers and user representatives think about barriers to effective knowledge sharing in Agile development.
	The impacts of Agile and lean practices on project constraints: a tertiary study	A tertiary study with 13 secondary studies is presented in this paper. It discusses how Agile and lean practices have their impact on projects. It also indicates that TDD has positive impact on external quality.
	Agile Methodologies in Software Maintenance: A Systematic Review	This paper presents a systematic review of 30 research papers between the years 2001 and 2015. It talks about the use of Agile in software maintenance in order to increase software quality.
2015	The Kanban approach, between agility and leanness: a systematic review	A systematic review of Kanban is presented in this paper. A total of 37 primary studies were selected and there are 20 different elements of Kanban that are considered and reported.
	Achieving agility through BRIDGE process model : an approach to integrate the Agile and disciplined software development	The main goal of this paper is to show that agility can also be achieved by traditional development process. This paper uses the BRIDGE model to depict the same. It integrates the traditional and Agile software development and establishes compatibility between these approaches.
	A systematic review of distributed Agile software engineering	Distributed human resources in Agile come with a number of challenges that needs to be considered and mitigated. Firstly, this paper talks about the conditions that lead to adopting Distribute Agile Software Engineering (DASE). Secondly, it talks about the risks associated to DASE and the strategies that exist to mitigate these risks. Lastly, it talks about the approaches that have been successfully adopted by the organizations.
2013	Evaluating the impacts of an Agile transformation: a longitudinal case study in a distributed context	This paper talks about what impact the introduction of Agile practices had in large software development organizations. The study concluded with two results. First, Agile practices has beneficial effects that were expected and second, with such a longitudinal study, it is possible to evaluate both the impact of Agile and its effects at very early stages in an organization.
2012	Agile software development for medium and large projects	Despite several benefits, there are few limitations to XP which are weak documentation, lack of strong architecture, ignorance of risk awareness and inappropriateness for medium and large projects. An extended XP model has been proposed in this paper which promises to provide equal benefits for medium and large projects like that of its benefits in small projects. Three industrial case studies are conducted to evaluate the proposed model. The results indicate that XP is equally beneficial to medium and large projects.

Only Journal articles were selected for this study. Table 3 below shows the publication channel and the number of articles related to that publication channel.

Table 3. Papers by Publishers

Publication channel	Number of papers
Springer	7
Informatica	1
Wiley online library	4
Journal of software engineering and development	1
International journal of electrical and computer science	1
Modern education and computer science	4
Institute of Engineering and Technology (IET) Software	1
Journal of information system education	1
Journal of system and software	2
Institute of Electrical and Electronic Engineers (IEEE)	1
International journal of multidiscipline science and engineering	1
International journal of advanced computer research	2

3. Research Question

This review has been done in order to address the strengths and weaknesses of traditional vs. Agile software development. Hence, the research question is can an ensemble method help in achieving a higher success rate in software development?

4. Related Works

According to Mandal *et al.* [16] the primary perceived weaknesses of traditional software development processes are as follows – excessive documentation, too sequential, excessive planning, a lack of results until the end, late communication to stakeholders, delays in project delivery and increased project costs. Agile methodologies were developed to overcome these weaknesses. Mandal *et al.* [16] also said that "Agile was a significant departure from the heavyweight document driven traditional software methodologies". There are 12 principles of Agile manifesto [1] whose detailed description is out of the scope of this paper. However, summarization of these principles was done by Mandal *et al.* [1] and are presented as follows – Customer satisfaction, incorporation of rapid system change, frequent working software delivery, continues corporation of client and developer, motivated trusted individuals, continuous improvement, arrangement of face-to-face conversation, progress measurement, sustainable development, attention to technical excellence, simplicity, self-organizing teams, internal assessment for knowledge enhancement, quality assurance and economic development.

The meaning of these principles is often misunderstood, wrongly interpreted and commercialized [1]. Many people claim to be Agile in the present software industry only because it is fashionable to be Agile [1]. There is a difference between "doing Agile" and "being Agile" [1]. Agile methods and practices are often poorly implemented [1]. There are still several organizations who are struggling to adopt Agile methods successfully [8, 10].

In spite of the fact that Agile has been accepted well by the software industry, it still has strengths and weakness which are highlighted by Tarwani et al. [25]. According to them, the strengths of Agile methodologies are early warning of risk, constant testing, iterations, small teams, customer feedback, quality, on time and on budget. Tarwani et al. [25] said that "the main strength of Agile due to which it had gained popularity over traditional and sequential waterfall model is that it is based on the concept of iterations". The weakness of Agile methodologies as presented by Tarwani et al. [25] are miscommunication, resource increase, overall cost increase, inappropriateness for large projects and lack of coordination.

Specifically talking about individual Agile methodologies, Scrum is one most of the widely used Agile methodologies. This methodology mainly relies on effective and efficient communication among the team members [22]. It is one of the best in management practices. The core values as stated by Qureshi in [22] are daily inspections, self-organized teams, the Scrum master, every sprint having a delivery and product owner setting priorities to the product backlog. According to [3], Scrum has short iterations which involve continuous feedback from the customer that makes it easy to cope with the changing needs and requirements of the industry. This in turn helps in delivering a quality product with customer satisfaction. Daily Scrum meetings and sprint meetings make it easier to measure the growth and productivity of a product and an individual working in that team. Also, testing is done at the end of each sprint which guarantees the quality and bugs are fixed right away. With several Scrum meetings, it becomes easier to follow the schedule and deliver the product on time. In spite of having several known strengths in Scrum methodology, it has its own weaknesses. Scrum software development is suitable for small projects and it becomes comparatively difficult to follow Scrum in large organizations. One of the main weaknesses of Scrum is reported in [23] that many employees in the software industry lack the knowledge of Scrum. They complete one simple course on Scrum and call themselves as Scrum masters. Another weakness is it lacks engineering practices [22] and so there were many attempts to combine Scrum with other methodologies; Scrum is simple to understand but difficult to master[19].

Another famous Agile methodology is eXtreme programming (XP). Like any other Agile methodology, XP also consists of iterations and in return gives quality products with customer satisfaction. XP can also handle unclear and changing requirements in the industry [5]. Many researchers have made attempts to integrate XP with Scrum or XP with any other methodology. Pair programming and continuous integration are the most used practices in XP and results in improving the productivity [25]. XP works well with simple and small scale projects and focuses more on coding than on the design [10]. The weaknesses of XP as stated in [5] are lack of documentations, poor architectural structure and less focus on design. There were 18 papers reviewed in [5] that showed that there were several attempts made to overcome these weaknesses. Some of them had solutions to a few weaknesses but not all. In [5], a simplified XP model has been proposed which overcomes all the weaknesses stated above but there is no validation done for the same. There are several studies which show that simply placing two programmers in front of the computer is not enough (pair programming in XP) [7]. Pair programming requires mutual understanding of both the programmers and a common skillset. It requires much knowledge and expertise of that domain by both programmers [5].

Lean is an Agile toolkit which has principles mainly focused on elimination of waste and maximization of value [3, 24]. The Lean methodology has been claimed as the fastest growing methodology for product development in the past decade [2]. It has a very behavioral approach [1]. However, Lean does not cover the technical and managerial issues. Its concerns are mostly about minimizing the wastage and hence improving the quality [3]. One of the most popular principles of Lean approach is Kanban [2]. Kanban is a visual method that helps in managing the production of a product [3]. This methodology can not only be used for development but also has its strengths in teaching, like used in [9]. With the usage of Kanban there is a positive increase in interaction and communication between the teams and stakeholders [9]. A total of 37 primary studies have been investigated in [2] which gives details about the strengths and weaknesses of Kanban. Although there are several definitions that have been defined in [18] for Kanban and lean, there is still lack of guidelines that say how both should be applied in software industry [2]. There is very limited research that gives guidelines in implementation of Kanban to the practitioners [2]. There is a strong need for systematic studies in the area of Lean and Kanban [2].

Table 4. Strengths and Weakness of Software Processes

Process	Strengths	Weaknesses
Scrum	Effective and efficient communication among team members One of the best management practices Continuous feedback from the customers Produces quality product with customer satisfaction Measuring the growth and productivity of the team and individual is easier with daily Scrum meetings and sprint meetings	Employees lack knowledge of Scrum Scrum lacks engineering practices Simple to understand but difficult to master Suitable for small projects
EXtreme programming	Quality product with customer satisfaction Can easily handle unclear and changing requirements Pair programming and continuous integration improves productivity Works well with simple and small scale projects	Lack of documentation Poor architectural structure Less focus on design Pair programming requires mutual understanding and common skillset between two programmers
Lean	Eliminate waste Maximize value of the product	Does not cover technical and managerial issues Lack of details about its implementation
Kanban	Helps in managing production of a product Increase in communication between the team and stalk holders	Lack of details about its implementation
Test driven development	Positive impact on external quality of the system Writes test cases and test code first using the requirements Writes Lean code, removes duplicates	Sometimes very time consuming due to repeated test failures Specific knowledge and skill set required
Crystal	Effective communication among team members Projects can be clearly classified using Crystal methods	Only two type of crystals are defined in details (Crystal clear and Crystal orange) Lacks system validation practices
Feature driven development	Adaptive and incremental in nature Emphasis more on quality	Needs special training to write requirement/user stories Less responsiveness to change Need of experienced and trained staff Less appropriate for small scale projects
Waterfall	Simple to understand and use [14] Each phase is clearly defined and well understood Detailed documentation [14]	Working software is delivered very late and hence it has lots of risks associated to it It is difficult to accommodate changes using waterfall [14] Measuring progress is difficult [14] Not suitable for projects with changing requirements [14]
Rational Unified Process (RUP)	Produces quality product Less time for integration Less development time	Needs expert team members trained in RUP Complex development process Development process is difficult to manage
Spiral	Requirements change is manageable Frequent delivered of working software Lower risk of failure [14]	Not suitable for small projects Process is difficult to manage Can continue indefinitely [14]

In a tertiary study done in [18], it has been revealed that test driven development (TDD) has a very positive impact on external quality of a product. The quality attribute included external quality, complexity, code size, etc. However, no conclusion was made on the impacts of TDD on code size. Test driven development is one of the most advantageous approaches that has produced several successful products because of its approach of writing the test cases and test code first by using the requirements. This reduces defect rate and improves quality of the product. It also helps in writing clean code and removing duplicates in each iteration [6]. Sometimes, TDD can be very time consuming due to repeated test failures [6]. Specific knowledge and special skills are required in order to implement TDD [6], [12].

Depending upon the size, complexity and team size, Crystal methodology can be used. There are several strengths and weaknesses about Crystal that are highlighted in [6]. Projects can be clearly classified using Crystal. It provides good risk control. However, out of four Crystal methods available, only two (Crystal clear and Crystal orange) of them are defined in detail. Life critical systems are difficult to develop using Crystal because it lacks system validation practices [6].

Like any other Agile methodologies, Feature driven development (FDD) has adaptive and incremental nature [17]. The emphasis of FDD is quality. It focuses more on designing and building aspects of the software development. As the name says, FDD has more focus on its feature development [6]. FDD does not provide any guidance about requirements gathering, analysis and risk management [6]. A simplified FDD process model was introduced in [17] to overcome its limitations (i.e., less responsiveness to changes, reliance on experienced staff and less appropriateness for small scale projects). There were 14 research papers that were discussed in [17]. Several processes were proposed in those papers but none overcame all of its limitations. However, this simplified FDD was not validated.

There are several methodologies that exist for software development. Each of these have their own strengths and shortcomings. No particular process exists that satisfies all the weaknesses of a project and gives the best result. A new process can be developed by integrating all these methodologies and core engineering practices so that this new process uses all of their strengths, overcomes the weaknesses of each other and yields the best results.

5. Conclusion and Future Work

In this study a systematic review of 25 papers were done. The findings from this study show that there does not exist a one-size-fits-all methodology in software development which does not have any limitation. Every methodology including Agile or traditional has its own limitation as shown in table IV. This systematic review discusses about several strengths and weaknesses of the two methodologies. Many organizations use a combination of processes. Usage of such combinations has helped the organizations overcome weaknesses of a single process. For future directions, this study suggests that a new process be worked upon which integrates all the simple and value added features of all the processes that were discussed in Section 4. By integrating the traditional and Agile with the core engineering design process a new process can be developed. The new process can also undergo double validation (i.e. validating it twice).

References

[1] Hohl, P., Klünder, J., Van, B. A., Lockard, R., Gifford, J., Münch, J., Stupperich, M., & Schneider, K. (2018). Back to the future: Origins and directions of the “Agile Manifesto” – Views of the originators. *Journal of Software Engineering Research and Development*.

[2] Ahmad, M. O., Dennehy, D., Conboy, K., & Oivo, M. (2018). Kanban in software engineering: A systematic mapping study. *Journal of Systems and Software*, *137*, 96–113.

[3] Al-Baik, O., & Miller, J. (2015). The kanban approach, between agility and leanness: A systematic review. *Empirical Software Engineering*.

[4] Albarqi, A. A., & Qureshi, R. (2018). The proposed l-scrumban methodology to improve the efficiency of agile software development. *International Journal of Information Engineering and Electronic Business*.

[5] Korhonen, K. (2013). Evaluating the impact of an agile transformation: A longitudinal case study in a

distributed context. *Software Quality Journal*.

[6] Anwer, F., & Aftab, S. (2017). SXP: Simplified extreme programing process model. *International Journal of Modern Education and Computer Science*.

[7] Anwer, F., Aftab, S., Waheed, U., & Muhammad, S. S. (2017). Agile software development models TDD. *FDD, DSDM, and Crystal Methods: A Survey*.

[8] Chen, K., & Rea, A. (2018). Do pair programming approaches transcend coding? measuring agile attitudes in diverse information systems courses. *Journal of Information Systems Education; West Lafayette*, *29(2)*, 53–64.

[9] Ghobadi, S., & Mathiassen, L. (2016). Perceived barriers to effective knowledge sharing in agile software teams. *Information Systems Journal*.

[10] Harzl, A. (2017). Can FOSS projects benefit from integrating Kanban: A case study. *Journal of Internet Services and Applications*.

[11] Heikkilä, V. T., Paasivaara, M., Lasssenius, C., & Damian, D. (2017). Managing the requirements flow from strategy to release in large-scale Agile development: a case study at Ericsson. *Empirical Software Engineering*.

[12] Karac, I., & Turhan, B. (2018). What do we (really) know about test-driven development? *IEEE Software*.

[13] Kaushik, S., Bharadwaj, A., Awasthi, V., & Sharma, R. (2017). Applicability and issues in traditional model of ERP implementations: an industry perspective. *International Journal of Advanced Computer* Research.

[14] Kazim, A. (2017). A study of software development life cycle process models. *International Journal of* Advanced *Research in Computer Science; Udaipur*.

[15] Kulkarni, R. H., Padmanabham, P., Harshe, M., Baseer, K. K., & Patil, P. (2017). Investigating agile adaptation for project development. *International Journal of Electrical and Computer Engineering; Yogyakarta*.

[16] Mandal, A., & Pal, S. C. (2015). Achieving agility through BRIDGE process model: An approach to integrate the Agile and disciplined software development. *Innovations in Systems and Software Engineering*.

[17] Nawaz, Z., Aftab, S., & Anwer, F. (2017). Simplified FDD process model. *International Journal of Modern Education and Computer Science*.

[18] Nurdiani, I., Borstler, J., & Fricker, S. A. (2016). The impacts of Agile and lean practices on project constraints: A tertiary study. *Journal of Systems and Software*..

[19] Perkusich, M., Gorgônio, K. C., Almeida, H., & Perkusich, A. (2017). Assisting the continuous improvement of *Scrum* projects using metrics and Bayesian networks: Assisting the continuous improvement of *Scrum* projects using metrics and Bayesian networks. *Journal of Software: Evolution and Process*.

[20] Poth, A., Sasabe, S., Mas, A., & Mesquida, A. L. (2019). Lean and Agile software process improvement in traditional and Agile environments. *Journal of Software: Evolution and Process*.

[21] Qureshi, M. R. J. (2012). Agile software development methodology for medium and large projects. *IET Software*.

[22] Qureshi, M. R. J. (2017). Evaluating the quality of proposed agile XScrum model. *International Journal of* Modern *Education and Computer Science*.

[23] Rizvi, B., Bagheri, E., & Gasevic, D. (2015). A systematic review of distributed Agile software engineering. *Journal of Software: Evolution and Process*.

[24] Santos, P. S. M. D. (2018). Link to external site, this link will open in a new window. *Journal of Software*

Engineering Research and Development.
[25] Tarwani, S., & Chug, A. (2016). Agile methodologies in software maintenance: A systematic review. *Informatica; Ljubljana*.

Mahrukh Sameen Mirza is a graduate student at University of Houston, Clearlake (UHCL) in Houston, Texas, USA. She has completed her bachelors in computer science in the year 2016 from Osmania University, Hyderabad, India. Her research interests are agile software development, data mining, artificial intelligence and software testing. She is currently doing her research in agile software development.

Soma Datta joined University of Houston Clear Lake (UHCL) as assistant professor in software engineering in the College of Science and Engineering. She received her Ph.D. in computer science from Texas Tech University. Her research interest are in data mining, developing pedagogies to teach engineering to middle school, undergraduate, and graduate classes for better concept retention. Her teaching interests include software processes, data science and R in software engineering, testing, verification and validation, agile software development, software engineering tools, reuse and reengineering, fundamental software development, introduction to engineering.