Proceedings of the Pakistan Academy of Sciences:
A: Physical and Computational Sciences 58(1): 17–33 (2021)
Copyright © Pakistan Academy of Sciences
ISSN (print) 2518-4245; ISSN (Online) 2518-4253
https://doi.org/10.53560/PPASA(58-1)721

Pakistan Academy of Sciences

Review Article

Agile Software Development Techniques: A Survey

Marriam Nawaz[1], Tahira Nazir[1], Seema Islam[2], Momina Masood[1], Awais Mehmood[1], and Samira Kanwal

[1]Department of Computer Science, University of Engineering and Technology, Taxila, Pakistan.
[2]Department of Computer Science, Comsats University Islamabad, Taxila Campus, Pakistan,

Abstract: In this IT era, where there is a race of software development, it is necessary to introduce such types of software development techniques which will help the practitioners to deliver fast solutions. In the past, various traditional approaches were used for this purpose, but now agile techniques are getting more popular because conventional software development approaches are not efficient in managing the changing requirements. The agile software development process is one of the most emerging lightweight software development methodologies, which uses iterative and prototype development approaches to accommodate changes in software requirements. Final software products are delivered to the end-users in short iterations. One of the most noticeable drawbacks of agile methods is their limited courtesy to the structured and architectural design of the system. Hence this development approach will restrict small to medium design decisions only. In this paper, we have performed the analysis of different agile techniques, which will help the readers to understand their positive and negative points and select the most appropriate technique suited to their projects.

Keywords: Agile Techniques, Fast Software Development, Survey, Traditional Development Approaches.

1. INTRODUCTION

It is an organization that thrives on delivering products faster, better, and cheaper ways. Many studies and suggestions have been conducted for the improvement of the software development process. Recently a new software development method has been introduced called Agile Software Development. Agile software development methods are introduced to tackle fast changes in organizational and business needs. Agile methods aid in generating quicker, faster, and efficient solutions. There is a huge difference between ASD and traditional approaches as ASD has more emphasis on a mechanism for change management during project development. In contrast, the traditional approaches have more focus on up-front and strict plan-based control.

The agile software development model is one of the major models of software development that is used extensively by industries, and huge research work is conducted on its methodologies nowadays [1-4]. This approach is used as an alternative to conventional methods of software development as they are document-driven and heavyweight software development processes [5, 6].

Traditional approaches used for software development consist of several phases where for each phase, there is a predefined outcome and target [7]. But this caused a lot of problems like failure of software projects, unable to respond to changing requirement [5, 8] and also piles of documents gathered at the end of project development, But as requirements get changed many times throughout the project, so most of the time we do not require these documents as they are useless, So to cope with these problems Agile Software development model was introduced [9].

The agile software development methodology is based on the idea that software requirements are changing during the whole development lifecycle

Received: September 2020; Accepted: March 2021
*Corresponding Author: Tahira Nazir <tahira.nazir77@gmail.com>

[10, 11]. This approach provides a consistent way to deal with this dynamic behaviour of requirements as Process ability to iterate itself, having active interaction and communication among clients and development team, flexibility in project management and active involvement of customer during the whole development cycle are the main characteristics of agile software development [12, 13]. Another basic characteristic of the agile method is its provision of communication both among the development team and customers. The word "communication" has a very strong impact in the field of software development as it depicts that people who are working on the same projects will be agreed to the same standards, definition and will share their knowledge, provide information to others and have good coordination in their activities [14, 15]. So this will help to achieve its goal and result in customer satisfaction [16, 17]. Some examples of these practices are collaboration activities like scrum meetings which are held on a daily basis, pair programming, and having face-to-face discussions instead of using formal documentation methods [18, 19]. So as communication is the central property of agile methods and makes it distinct from other traditional approaches [20, 21].

Agile software development techniques are preferred to use in such an environment where there is a chance of sudden change or have to generate a quick reaction to changing requirements by delivering small increments or through continuous incorporation of customers [22, 23]. Several principles of the agile method exist, of which some are based on behavioural and some are based on managerial improvement for software development [24, 25]. Agile software development methodologies are mainly concerned with code development rather than documents driven [26, 27].

There exist several agile software development methods that promote development work, collaboration among team members, and increase the flexibility of processes to make them more adaptable throughout the development lifecycle [28]. These methods include XP (Beck, 1999), [29], FDD (Feature-Driven Development) (Palmer & Felting, 2002) [30], Scrum methodology (K. Schwaber & Beedle, 2002), [31], ASP (Adaptive Software Development (Highsmith, 2000)[32], and DSD (Dynamic Systems Development Method) (Stapleton, 1997) [33]. XP and Scrum are considered as the best agile software development methods[34, 35]. The main focus of Scrum is on software project management to increase their probability of success while XP is more concerned with project-level activities of software development [36]. All agile software development approaches (Scrum, XP, DSD, ASP, FDD, RUP) are iterative and Incremental and have focused on different parts of the software development lifecycle. Among them, some approaches have focused on different practices used for development like XP, Agile Modelling, and pragmatic programming while other concerns with software project management like Scrum approach [37-39].

This study is focused on a comparative analysis of agile software development techniques and their current practices in the industry. These approaches will be examined from the angle of their applicability, strengths, weaknesses, product delivery, standards used for coding, design standards, roles description, and complexity of design and workflow technique. This will lead the reader to find benefits, limitations, and difficulties in the transition from traditional to agile software development. Moreover, this paper explains the worth of employing agile techniques in software development by examining its various methods. The presented research work demonstrates that agile approaches have significant benefits as compared to the existing traditional methods. However, all benefits do not apply to all software projects and situations.

The rest of the paper is divided into the following sections: section 2 explains the traditional approach that is waterfall method for software development, section 3 contains a description of agile software method and its comparison with waterfall method. Further, this section gives a comparative analysis of agile software development methods and section 4 comprises of conclusion.

2. WATERFALL MODEL

The waterfall model is the first traditional model to be introduced. It is a static technique that linearly performs the software development. This approach is very simple to understand and completes one activity before starting another. The waterfall model divides the projects based on process activities like

planning, design, implementation, etc [35]. There is a predefined goal for each development phase. The first phase must be completed before going into the next phase and there is no way to go to the previous one [40]. Testing can only be performed when the whole project is completed [39].

This approach applies to those systems that are more structured and where there exists a small chance of modification after development [40, 41]. It is difficult to reuse and upgrade the software systems developed by using this technique because there exists a coupling between data and code[41]. So if data is changed then code must be modified according to it and this causes to increase the overall cost of a project because the whole process needs to be modified [42, 43].

Figure 1 shows the workflow of the waterfall model:

3. AGILE SOFTWARE DEVELOPMENT

The word Agile states 'moving fast' or 'quickly accepts changes'. It is a lightweight and practice-based technique used widely for software development nowadays. It understands and accepts the idea that handling each project varies from each other, so a more dynamic approach for modeling is required that can be tailored according to the needs of different projects [42, 43].

Instead of following a single long process for the development of projects Agile methodology divides the development cycle into small chunks called increments [44]. After completion of each increment, it is delivered to the users for their verification. It follows the iterative approach and the final product contains all required features of users [45]. Figure 2 presents a graphical representation of the Agile Methodology. Table.1 shows the basic principles of the agile technique[47]. Table.2 shows the comparison between the traditional approach and agile methodology.

3.1 Extreme Programming (XP)

XP is one of the first agile methodologies which are proposed to improve the quality of software. It is a lightweight technique that provides a quick response to the evolving requirements of users and supports a more iterative and well-planned method of software development. It contains a small team of developers and provides an intense level of interaction between the development team and client organization in the whole development

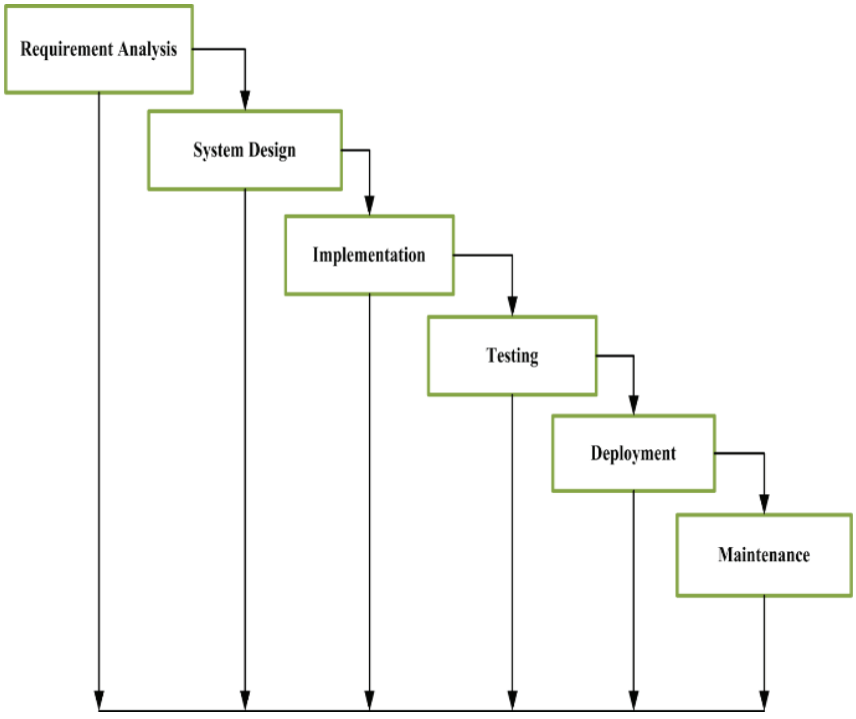

Fig. 1. Waterfall Model Development Lifecycle [41].

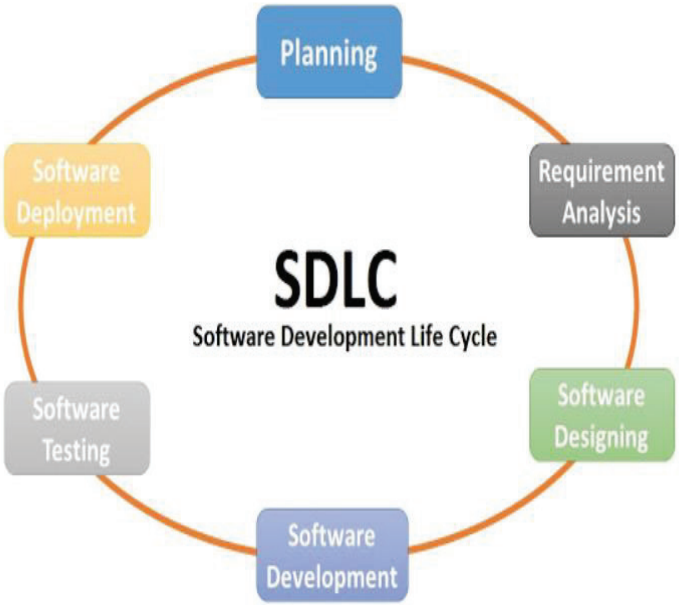

Fig. 2. Agile software development lifecycle [46].

lifecycle.

Being a type of agile methodology it provides fast "releases" in small cycles of development which not only results in increasing the production rate but also provides the points where evolving requirements of customers can be facilitated [48]. The name for this methodology comes from the idea of taking the different elements of traditional approaches to "extreme level". It addresses the different phases of the software development lifecycle like analysis, design, implementation, and testing phases with novel techniques that will cause to raise the quality of end product [49].

The basic principle of XP is to organize the people in such a way to improve the quality of end products and reduce the cost of accommodating the varying requirements of users by following multiple small phases of development [50]. Figure.3 explains the core practices which are used in XP:

Distinguishes features of XP [51] are as follows:

- Story Cards: Users define requirements as story-type scenarios, which are then presented in the form of story cards. Each story card is then further divided by developers to break them into smaller tasks. These smaller tasks are then prioritized with the help of customers for implementation.
- Simplicity: XP works with designing the simplest product to meet the basic needs of users. It is based on the principle to only develop what is demanded in the given requirement. Further functionalities are added to the product according to users' needs.
- Feedback: At the end of each release, proper feedback is obtained from the customers, and the next level of iteration is based on this feedback. In XP, for efficient feedback, small loops of design and implementation are built with the help of a pair programming technique and a test-oriented development method.
- Test-Driven Development: Extreme programming uses a test-oriented development technique in which test cases are pre-written before actual code implementation. Testing is used throughout the process of XP.
- Refactoring: It always encourages finding the

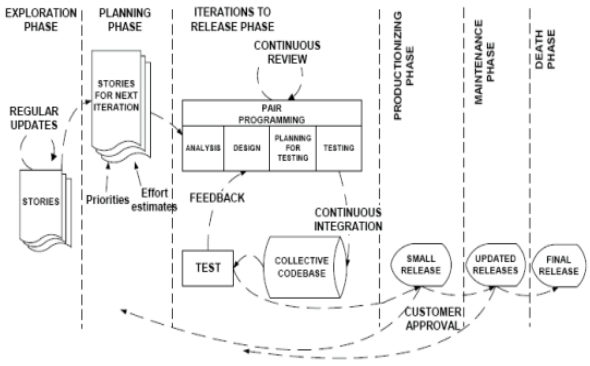

Fig. 3. XP workflow [45].

Table 1. Basic principles of agile methods

1	The main objective is the satisfaction of customers through fast and early delivery of valuable features.
2	No matter which phase of development you are, it must be able to accept and accommodate the changes.
3	Increment should be delivered quickly within weeks or months.
4	Strong communication between the development team and the customer organization.
5	Must be able to provide sustainable development to every stakeholder, whether he is a developer, customer, or sponsor, so that he has a constant pace.
6	The whole team should participate in identifying the ways of becoming more effective and then model their behavior according to this.
7	Involve trusted and motivated individuals in projects.
8	Continuous attention to technical excellence and good design.
9	Simplicity—the art of maximizing the amount of work not done—is essential.
10	The basic measure used for progress checking is working software.
11	The team should be self-organizing to select the best technique for requirement gathering, design, and architectures.
12	Both the development team and customer organization should work closely throughout the development lifecycle.

Table.2 Shows the comparison between the traditional approach and agile methodology

Factors	**Traditional Development**	**Agile Development**
Development Process	Linear	Iterative
Development style	Analytical	Adaptive
Development Orientation	Process-Oriented	People-Oriented
Requirements	Complete understanding of requirements and should be documented and stable.	Discover with the progress of the project. Emergent and rapidly changed.
Project Type	Suitable for large project size.	Suitable for small or medium project size.
Planning Scale	long-standing	short-standing
Style of management	More controlled and command-oriented.	More collaborative and leadership-oriented.
Documentation produced	High	Small
Response to change	Resistive	Accepted and adaptive
Client interaction	Low	High
Team Organization	Structured	Self-organized
Success measure	Plan conformance	Delivering business value

best practices for both design and problem solutions and using them to modify the existing solutions. This will cause to improve the quality of the product.

- Pair Programming: Pair programming is the distinguishing feature of XP, where a pair of programmers works dynamically. This results in immense savage of time and reduces the working load.

The main benefit of using this technique is that it is speeding up the process of development as this approach gives the right to the developer to fix a fault in code when it is detected. Standards related to development and designs are defined globally so that the whole team follows the same conventions. This technique is suitable for small size applications that do not need proper planning and specification efforts [52]. It results in cost reduction because it does not include useless documentation and help

the developers to concentrate on their basic task and performs better risk management. As simplicity is an important feature of XP so it creates more high-quality and faster products and contributes a lot in increasing the robustness of products. At the same time, the main limitation of this technique is that it does not take into count planning or measuring Quality Assurance of design and coding [53]. As it involves pair programming, so there is a huge chance of duplication of data. And it is a code-centric technique and can be irritated in large projects.

3.2 Feature-Driven Development

Software features are the basic focus of this approach because these features are the main driver of the whole development lifecycle [54]. This method is different from other techniques of agile development because the planning of the whole project and upfront design is its basic concerns. It has a basic five stages [55].

3.2.1 Develop an Overall Model

FDD approach is different from XP and Scrum because it demands team effort at the beginning of the project for completely understanding the main structure of the problem under consideration by developing its object model. The basic reason for building this model is to get a good idea and a shared understanding of the project. It captures the following things:

- Requirements of users
- Assumptions of users

3.2.2 Build a Feature List

Based on the first activity, a list of features is defined in this phase. Functional requirements are divided into smaller activities where each activity will deliver some business value to users.

3.2.3 Plan by Feature

A complete formal team is involved in this phase which consists of a project manager, head of the development team, and chief programmer. A complete plan is prepared here to determine the order in which features will be developed. The plan is prepared based on the priorities of the customer, dependencies between modules, risk, and complexities. Completion dates are also finalized here.

3.2.4 Design by Feature

All design packages like sequential diagrams class diagrams are defined here by the chief programmer. The sequential diagrams are developed by a group of people, but class diagrams and object models are defined and developed by owners of the class. Feature requirements are modified here with the help of domain experts.

3.2.5 Build by Feature

Here all classes and methods which are outlined and designed in the design phase are practically developed by developers and are checked and inspected for defects by using unit testing. Figure 4 shows the lifecycle of FDD.

The implementation work of all features is performed in parallel and each team has its owner which makes it distinct from XP. This approach is well suited to the projects of large size and five stages of the process allow you to perform the work in a better and disciplined manner [57]. It uses a predefined standard for implementation of the project, so it makes work easier for developers [58]. This technique does not perform well for small team sizes and the success of the project is dependent on chief programmers [59]. No documentation is available in written form in this methodology.

3.3 Scrum

Scrum is one of the iterative agile software

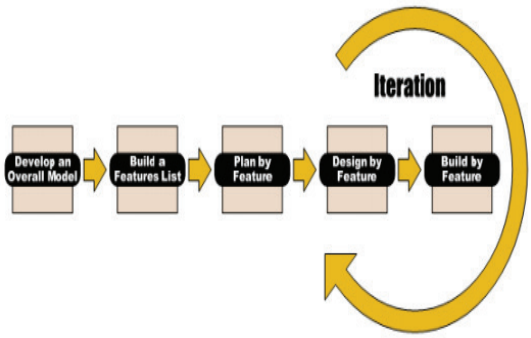

Fig. 4. FDD workflow [56].

development technique which is used for management of software product development [60, 61]. The main principle of this approach is to enable the development team to work as a unit and achieve common goals of the organization, enable the development team to self-organize and work at physical co-location where discipline and face to face communication of all team members are involved [62]. It is the responsibility of the scrum team to define organizational goals and then give their best to meet them.

3.3.1 Documents and Artifacts

Scrum team generally produced three main documents and artifacts these are Sprint Burndown chart, the Sprint Backlog, and the Product Backlog.

3.3.2 Sprint Burndown Chart

Burndown chart is one of the most common mechanisms for sprint tracking used by the scrum team. Burndown Chart is a graphical representation of time versus work left to do, time is often at a horizontal axis and work remaining on the vertical axis.

3.3.3 Sprint Backlog

A sprint is a list of all possible business and technology attributes and a list of all errors and defects that have to be managed and scheduled for the iteration on which we are currently working. The spreadsheet is used for defining Sprint Backlog. In which requirements are represented as tasks. The spreadsheet consists of a short task description region for each task. On basic daily spring, the backlog is updated by a daily tracker that keeps the latest estimate of work complete vs work remaining to complete.

3.3.4 Product Backlog

It is the prioritized queue of all technical functionalities that need to be developed by the development team and evaluate all the defects that need to be fixed. A unique identifier or ID is assigned for each requirement in the product backlog. Product Backlog is also kept in a spreadsheet. An overview of the whole process is explained in Figure 5.

The main power of this technique is that it conducts the meetings on daily purposes to keep the team focused and save both time and money Regular communication and interaction between SCRUM team members helps in attaining efficient completion [63]. In the SCRUM process, frequent testing is conducted which ensures that development work is going well. Regular feedback means changes can easily be tackled before the project grows too large [64]. This technique works well with small teams and can be inefficient due to slacking team members. Sometimes team member is not open to the flexibility it means that removal of one or two team members will cause disastrous damage to the whole team [65].

3.4 Dynamic Systems Development Method (DSDM)

The dynamic systems development approach is purely based on the development of such systems that focuses on the development of that business application whose purpose is to fulfill the needs of the business [67]. DSDM is an evolutionary development approach that uses the timebox and

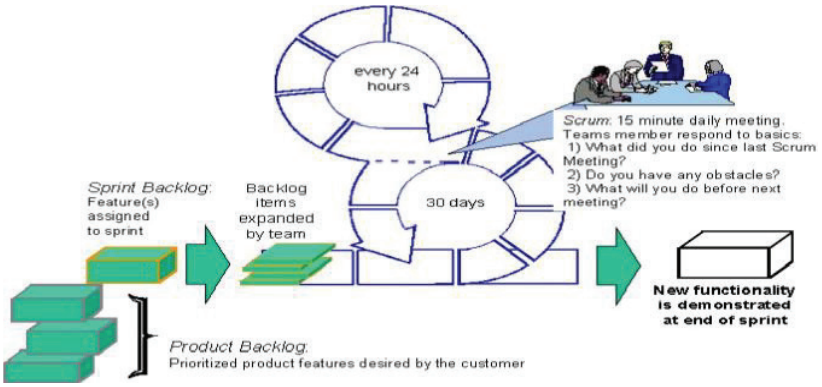

Fig. 5. Scrum workflow [66]

task prioritization approach. DSDM model has very strict standards and very inflexible deadlines for project completion. DSDM testing is an umbrella activity that occurs throughout the entire development life cycle. Feedback is gathered at each stage by the project team and project owner, who shared a physical or virtual workplace for efficient communication. DSDM works efficiently for large or medium-sized projects [68].
Implementation Build and design iteration, Functional Model iteration, Business study, and feasibility study are the few phases involved in DSD methodology.

3.4.1 Feasibility Study

In this stage feasibility report is generated, it is judged that either it will be suitable to develop a product with DSDM or not. Risk and other technical issues are also explored during this phase.

3.4.2 Business Study

System architecture and product outline are prepared at this phase. In this phase, primary business and technical information are studied, the process is defined according to business needs and requirements.

3.4.3 Functional Model Iteration

This is the iterative stage where the actual development starts, at the end of this phase, code prototype and analysis model are prepared.

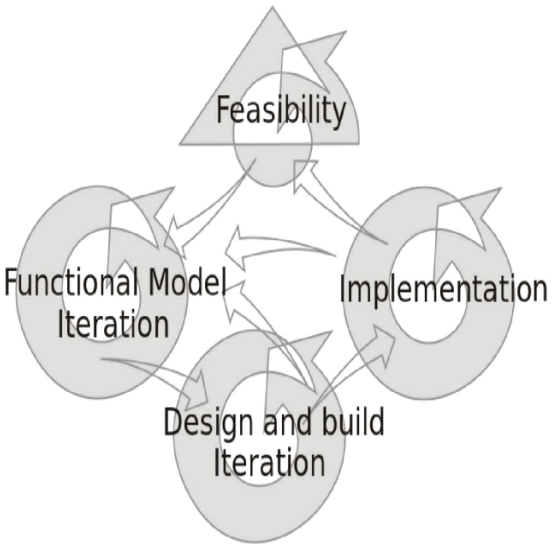

Fig. 6. Dynamic System Development Method Work-flow [69].

3.4.4 Build and Design Iteration

This is an iterative phase where customer requirements are evaluated, and direct communication is conducted with users to know that if end-users need further changes in development or not.

3.4.5 Implementation

This is also an iterative phase in which a completely implemented product is handover to the customers.

In this approach, Users get a stronghold of the software development process. As deadlines are un-flexible so quick delivery of functionality is possible [70]. But this technique is very costly to implement and for the small organization, this method is not suitable. If a user is not a domain expert, then the involvement of the user may be dangerous.

3.5 Crystal Methods Agile Software Development

Alistair Cockburn developed a crystal family (family of methodologies) [71]. Crystal methods are considered "lightweight software development methods" [72]. Cockburn [71] differentiates methodologies, techniques, and policies as follows:

Policies: Organizational standards or roles and regulations.
Techniques: Areas of expertise
Methodology: Practical tools

The Crystal family of methodologies assign a different color to different methods according to their "weight". Crystal yellow, crystal orange, or crystal-clear methods will be used if projects are small one. For safety-critical systems, crystal diamond or crystal sapphire will be used [37]. Crystal family has divided into the following colors: i) Crystal Clear ii) Crystal Yellow iii) Crystal Orange iv) Crystal Orange Web v) Crystal Red vi) Crystal Maroon vii) Crystal Diamond viii) Crystal Sapphire.

Some of the basic properties of the crystal family are as follows:

3.5.1 Frequent Delivery

Frequent delivery of software products by iterative development of the system. By releasing the product in iteration, end users can early identify the problems, and this then allows developers to tackle the problem earlier and ultimately will reduce the time and cost for re-development of a software system.

3.5.2 Reflective Improvement

In this approach, developers take a break from regular software development and explore new ways in which they can better develop software systems, feedback is taken at each iteration for further improvement.

3.5.3 Personal Safety

All people in the team should be allowed to speak freely about their ideas and suggestions, No one should be ridicule otherwise, they will be less likely to speak next time and overall team communication will be affected.

3.5.4 Easy Access to Expert Users

The developer will work with domain expert individuals, the greater the involvement of expert users the greater will be the chance of better product development.

Crystal family methods are suitable for small to very large projects. Face to face communication, consider talents, people, and community are the main aspect of these methods. But these approaches are not suited for medium-sized systems. Customer's unavailability can also degrade the performance of these methods.

3.6 Lean-Agile Software Development

Among all other agile software development techniques, the lean-agile methods are one of the most strategically focused methods. The main goal of this method is to develop the software system in one-third of the time with less budget and less amount of workflow.

Basic principles of LEAN agile software development are as follows [73]:

- Eliminate waste
- Respect people
- Optimize the whole
- Build quality
- Deliver fast
- Defer containment
- Create knowledge

By following this technique, the cost of the software development system will potentially reduce if elimination of overall efficiency is done earlier [74, 75]. It results in the early delivery of software systems and the efficient decision-making ability of the software development team [76]. The workflow of Lean agile software development is given in Figure.7.

3.7 Agile Modeling

This technique is used for documenting and modelling the software-based system by selecting an approach based on best practices. It consists of different values, practices, and principles which are used for documenting and modelling different software systems. This approach is more flexible and easy to practice as compared to traditional approaches [78]. The main objective of this technique is to document the systems by keeping its amount as low as, it is possible [79]. Different types of cultural issues exist, but they are resolved by encouraging and providing proper communication among team members [80]. This technique is used as an addition to other approaches of agile development like Scrum, XP, etc [81]. Figure 8 shows the lifecycle of agile modeling.

This approach helps to better maintain the significant documentation of the system. It provides a better resolution of cultural issues by providing good communication among team members [81, 83]. But it cannot provide a good result with poor modelling techniques and complex with large team size if proper tooling support is not available.

3.8 Adaptive Software Development

The adaptive software development (ASD) technique has emerged from the rapid application development approach. Different phases of this technique like speculate, collaborate, and learn are introduced to replace the traditional approaches used for software development [84]. These

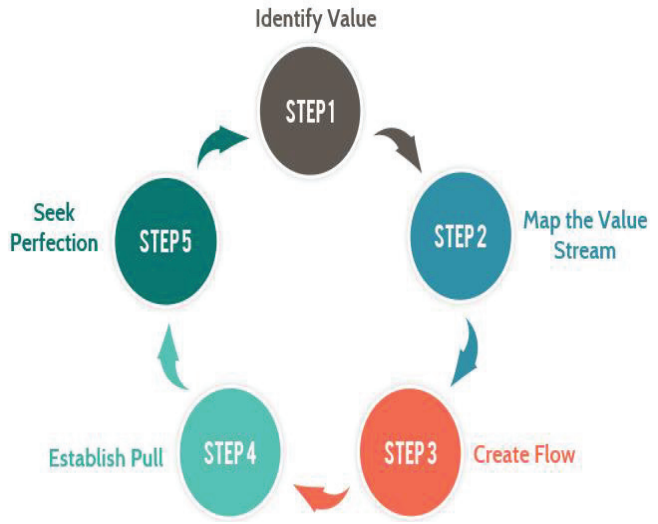

Fig. 7. Lean-Agile Software Development[77]

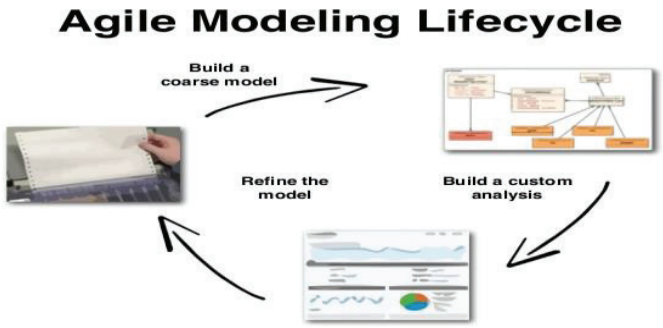

Fig. 8. Agile modelling lifecycle [82].

techniques are adaptable and can accommodate the changes easily in an unstable environment. Mission-focused, iterative in nature, provides tolerance to change, feature-based approach, and risk driven are basic characteristics of ASD [85, 86].

ASD consists of three phases:

3.8.1 Speculate

In this phase, the project is initiated, and all risk-driven plans are developed here. The basic motive of this phase is to completely understand the requirements of users so that the programmer can develop an understanding of the nature of the system under consideration. The success of this phase depends on bug identification and user reports for better guiding the project.

3.8.2 Collaboration

The parallel development of different components is performed in this phase. Proper customer and team collaboration are very important for the successful execution of this step which requires effective communication, creativity, and co-operated teamwork. For efficient requirements gathering JAD (joint application development) approach is preferred here. Instead of getting information about design details, code structure, or testing techniques 'collaboration' among developing team and client organization is the basic concern of this phase.

3.8.3 Learning

In this phase, all quality-related reviews are performed, and the newly created version of the project is made visible to users outside of the

Fig. 9. Adaptive software development Cycle [87].

development organization. Several bugs and user reports are produced here. Component's testing is performed thoroughly here. Figure 9 presents the flow of Adaptive software development.

All these phases show the dynamic and evolving nature of ASD which has replaced determinism with emergence [88]. This method is good to change adaption but as there is a fixed time of development so much pressure on the development team.

3.9 Kanban

This approach is gaining popularity in the field of software development. It provides a way to show and limit the progress of work during the development lifecycle. Its main focus is on doing proper scheduling of work so that product is timely delivered to customer organization [89, 90]. So the Kanban approach is responsible for the management of product development by ensuring its continual delivery to users without having to put a burden on the development team [91, 92]. Figure 10 shows the workflow of the Kanban methodology:

Distinguishes features of Kanban methodology are as follows:

3.9.1 Kanban Board

It is a tool used for visualizing the workflow of the project. It divides the work into different categories which are as follows:

- Backlog
- To-do
- In progress
- Done

3.9.2 Maximizes Productivity

By dividing the work into different groups this approach results in optimizing the workflow. It increases team productivity by minimizing idle time.

3.9.3 Continuous Delivery

This methodology is based on the continual releases of software increments rather than delivering the batches of functionalities.

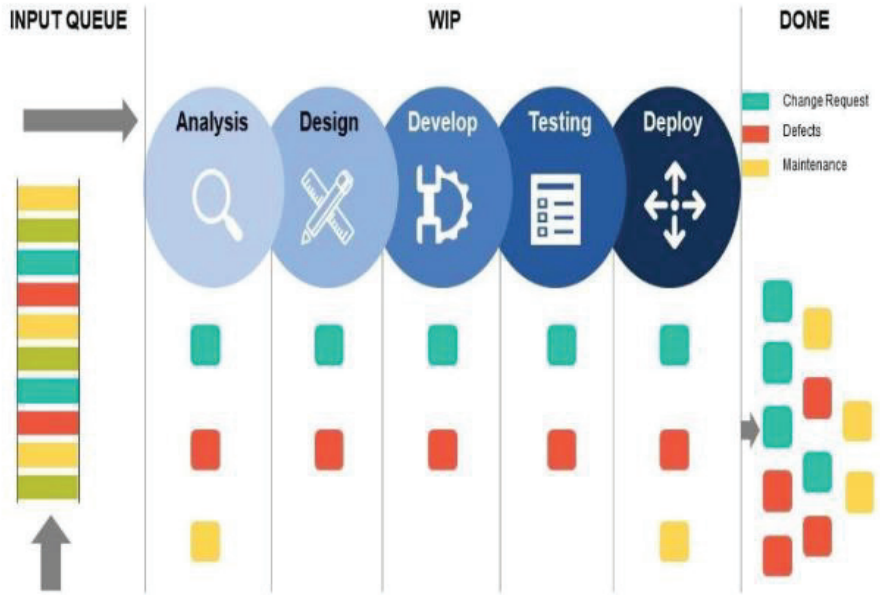

Fig. 10. Kanban workflow[93].

Table 3. Comparative analysis of agile methods in term of process

Factors	Scrum Methodology	Extreme Programming	Feature-Driven Development	Kanban Approach	Dynamic System Development System
Design Standards	Use complex design principles.	Use simple design and coding standards.	Use simple design approaches.	Guaranteed to reduce the waste by limiting the work in progress.	independent Framework for developing and implementation
Roles Description	Roles are predefined.	Roles are not predefined.	Roles are predefined.	Roles are predefined.	Roles are predefined.
Complexity of Design	Design complexity is high	Low design complexity.	Low design complexity.	Simple design.	Simple design
Workflow Technique	Work in iterations. Sprints are produced.	Does not work in iteration rather follow the task flow approach.	It is an incremental and iterative approach. A set of features is delivered.	Works in small iterations.	Iterative delivery of functionality.
Technique for Requirements Management	Product and sprint backlog is used for managing requirements in term of artifacts.	Story cards are used for requirement management.	Manage user requirements by building an object model of them	Kanban Boards are used for requirement management.	Timeboxing and Moscow principle is used.
Product Delivery Approach	Sprints are delivered on a defined time.	Continuous Delivery	Continuous Delivery	Continuous Delivery	Continuous Delivery
Standards used for coding	Coding standards are not defined.	Use defined coding standards.	Development practices are defined in advance.	Coding standards are not defined.	Coding standards are not defined.
Testing techniques	No formal techniques are defined for performing testing.	Use several testing techniques for product auditing like acceptance testing.	Use standard testing techniques.	At the end of each increment or work product testing is performed thoroughly.	Use standard testing techniques.
Changes acceptance	Changes are not acceptable in sprints.	It can accept changes at any phase of development.	It can accept the changing requirements of customers easily at all levels.	I can accept the changes at any time.	I can accept the changes at any time.
Process Owner	Scrum Master	Team Ownership.	Each class is owned by a class owner who works under a chief programmer.	Team ownership	Team ownership

Table 4. Comparative analysis of agile methods

Factors	Scrum Methodology	Extreme Programming	Feature-Driven Development	Kanban Approach	Dynamic System Development System
Customer Involvement	The presence of a customer on-site is not essential.	On-site customer presence and interaction are compulsory.	For the early two phases of FDD, customer involvement is mandatory.	Not essential for the on-site availability of customers.	It is a vendor or a customer independent approach.
Project director	Scrum Master	Extreme programming Coach	Project Manager	Teamwork	Teamwork
Collaboration among Team	Cross-functional teams	Self-organized teams	Teamwork.	The team consist of specific resources	Teamwork.

3.9.4 Waste Minimization

In this technique, tasks are only performed when they are needed. So, this approach results in avoiding over-production and wastage of time. Therefore, this approach is time-efficient.

3.9.5 Limits Work in Progress

Limiting the work in progress is the basic focus of this technique which optimizes the workflow of the system according to its capacity level.

4. COMPARATIVE ANALYSIS

In the presented work, we have discussed the detail of different agile development techniques. Moreover, the features and various phases of all approaches are also described.

In this section, we have presented a detailed comparison of agile techniques according to 3 p's (people, process, and product) of project management. Table 3 shows the comparison of all agile approaches concerning the process and Table.4 demonstrates the assessment of agile techniques from the perspective of people involved. After a detailed discussion of all agile techniques, we have selected different factors, which are used to perform the comparison of all approaches And for the third 'P' of project management, that is 'Product', it has been found that all agile methods apply to products of small and medium sizes.

5. RESULTS AND DISCUSSION

Software development approaches have been changing since the 1970s. To overcome the problems of traditional approaches, agile methodologies are introduced as these are lightweight in nature and help in accommodating the changes easily. In this paper, we present a comparison between traditional approaches and agile techniques used for software development. A comparison of agile methodologies is also performed in detail to highlight their various aspects. This study will help the readers to make a sense out of numerous agile techniques and can decide on which method is most suited to their problem. A major drawback of agile techniques is their inability to be used for big projects.

6. REFERENCES

1. T. Dreesen, R. Linden, C. Meures, N. Schmidt, and C. Rosenkranz. Beyond the Border: *A Comparative Literature Review on Communication Practices for Agile Global Outsourced Software Development Projects. in System Sciences (HICSS), 2016 49th Hawaii International Conference on.* (2016).
2. A. Zaitsev, U. Gal, and B. Tan, Coordination Artifacts in Agile Software Development. *Information Organization.* 30(2): p. 100288. (2020).
3. K. Suryaatmaja, D. Wibisono, and A. Ghazali, The Missing Framework for Adaptation of Agile Software Development Projects, in *Eurasian Business Perspectives. Springer.* p. 113-127.(2019).
4. C. Baham and R. Hirschheim, Issues, Challenges, and a Proposed Theoretical Core of Agile Software Development Research. *Information Systems Journal.* (2021).
5. Y. I. Alzoubi and A. Q. Gill. Agile Global Software Development Communication Challenges: A Systematic Review. in *Pacific Asia Conference on Information Systems.* (2014).

6. M. Marinho, J. Noll, I. Richardson, and S. Beecham. *Plan-Driven Approaches Are Alive and Kicking in Agile Global Software Development. in 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM).* (2019).
7. Y. B. Leau, W. K. Loo, W. Y. Tham, and S. F. Tan. Software Development Life Cycle Agile Vs Traditional Approaches. in *International Conference on Information and Network Technology.* (2012).
8. W. A. Cram, Agile Development in Practice: Lessons from the Trenches. *J Information Systems Management.* 36(1): p. 2-14. (2019).
9. A. P. Veiga, Project Success in Agile Development Projects. *arXiv preprint arXiv:*1711.06851. (2017).
10. K. Petersen and C. Wohlin, A Comparison of Issues and Advantages in Agile and Incremental Development between State of the Art and an Industrial Case. *Journal of systems and software.* 82(9): p. 1479-1490. (2009).
11. M. A. Akbar, J. Sang, A. A. Khan, S. Mahmood, S. F. Qadri, H. Hu, and H. Xiang, Success Factors Influencing Requirements Change Management Process in Global Software Development. *Journal of Computer Languages.* 51: p. 112-130. (2019).
12. F. Almeida, Challenges in Migration from Waterfall to Agile Environments. *World.* 5(3): p. 39-49. (2017).
13. J. Karrenbauer, M. Wiesche, and H. Krcmar. *Understanding the Benefits of Agile Software Development in Regulated Environments. in 14th International Conference on Wirtschaftsinformatik.* (2019 of Conference).
14. M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, and J. Still, The Impact of Agile Practices on Communication in Software Development. *Empirical Software Engineering.* 13(3): p. 303-337. (2008).
15. R. Lal, K. Kusuma, and S. Richter. Agility Practices for Software Development: An Investigation of Agile Organization Concepts. in *30th Australasian Conference on Information Systems.* (2019).
16. V. Holzmann and I. Panizel. Communications Management in Scrum Projects. in *European Conference on Information Management and Evaluation.* (2013).
17. A. Jarzębowicz and P. Weichbroth, A Qualitative Study on Non-Functional Requirements in Agile Software Development. *IEEE Access.* 9: p. 40458-40475. (2021).
18. S. Bock and R. S. Pusch, Application of Agile Methodology in Human Performance Technology. 2017.
19. E. Lozada-Martinez, J. E. Naranjo, C. A. Garcia, D. M. Soria, O. R. Toscano, and M. V. Garcia. *Scrum and Extreme Programming Agile Model Approach for Virtual Training Environment Design. in 2019 IEEE Fourth Ecuador Technical Chapters Meeting (ETCM).* (2019).
20. M. Hummel, C. Rosenkranz, and R. Holten, The Role of Communication in Agile Systems Development. *Business & Information Systems Engineering.* 5(5): p. 343-355. (2013).
21. S. Abdullahi and L. I. Bagiwa, A Review on the Process of Adoptability of Agile Methods in Software Development Practices. *American Journal of Engineering Research.* p. 199-207. (2019).
22. Z. H. Malik, An Application of Agile Principles to the Systems Engineering Lifecycle Process. 2017, The George Washington University.
23. P. P. Joby, Exploring Devops: Challenges and Benefits. *Journal of Information Technology.* 1(01): p. 27-37. (2019).
24. A. A. Siqueira, S. Reinehr, and A. Malucelli. *Using a Statistical Method to Compare Agile and Waterfall Processes Performance.* in *European Conference on Software Process Improvement.* (2017).
25. T. Kamal, Q. Zhang, and M. A. Akbar, Toward Successful Agile Requirements Change Management Process in Global Software Development: A Client–Vendor Analysis. *IET Software.* (2019).
26. T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, A Decade of Agile Methodologies: Towards Explaining Agile Software Development. 2012, Elsevier.
27. M. Stadler, R. Vallon, M. Pazderka, and T. Grechenig, Agile Distributed Software Development in Nine Central European Teams: Challenges, Benefits, and Recommendations. *International Journal of Computer Science Information Technology* Vol. 11. (2019).
28. F. Kamei, G. Pinto, B. Cartaxo, and A. Vasconcelos. *On the Benefits/Limitations of Agile Software Development: An Interview Study with Brazilian Companies.* in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering.* (2017).
29. K. Beck, Embracing Change with Extreme Programming. *Computer.* 32(10): p. 70-77. (1999).
30. J. M. Felsing and S. R. Palmer, A Practical Guide to Feature-Driven Development. *IEEE Software.* 7: p. 67-72. (2002).

31. K. Schwaber and M. Beedle, *Agile Software Development with Scrum.* Vol. 1. Prentice Hall Upper Saddle River.(2002).
32. J. A. Highsmith, *Agile Software Development Ecosystems.* Addison-Wesley.(2002).
33. J. Stapleton, *Dsdm, Dynamic Systems Development Method: The Method in Practice.* Cambridge University Press.(1997).
34. O. Salo and P. Abrahamsson, Agile Methods in European Embedded Software Development Organisations: A Survey on the Actual Use and Usefulness of Extreme Programming and Scrum. *IET Software.* 2(1): p. 58-64. (2008).
35. Ö. Özcan-Top and F. McCaffery, To What Extent the Medical Device Software Regulations Can Be Achieved with Agile Software Development Methods? Xp—Dsdm—Scrum. *The Journal of Supercomputing.* 75(8): p. 5227-5260. (2019).
36. A. Moniruzzaman and D. S. A. Hossain, Comparative Study on Agile Software Development Methodologies. *arXiv preprint arXiv:*1307.3356. (2013).
37. P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, Agile Software Development Methods: Review and Analysis. *arXiv preprint arXiv:1709.08439.* (2017).
38. S. ATAWNEH, The Analysis of Current State of Agile Software Development. *Journal of Theoretical Applied Information Technology.* 97(22). (2019).
39. S. M. Saleh, S. M. Huq, and M. A. Rahman. *Comparative Study within Scrum, Kanban, Xp Focused on Their Practices.* in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE).* (2019).
40. R. Narayan, Study of Various Software Development Methodologies. *EPRA International Journal of Multidisciplinary Research.* (2021).
41. A. M. Dima and M. A. Maassen, From Waterfall to Agile Software: Development Models in the It Sector, 2006 to 2018. Impacts on Company Management. *Journal of International Studies.* 11(2): p. 315-326. (2018).
42. H. Sharp and T. Hall, *Agile Processes in Software Engineering and Extreme Programming.* Springer. (2016).
43. J. A. Highsmith and J. Highsmith, A*gile Software Development Ecosystems.* Addison-Wesley Professional.(2002).
44. A. Alashqur, Towards a Broader Adoption of Agile Software Development Methods. I*nternational Journal Of Advanced Computer Science And Applications.* 7(12): p. 94-98. (2016).
45. P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, Agile Software Development Methods: Review and Analysis. *arXiv preprint arXiv*:.08439. (2017).
46. B. O. Egho-Promise EhigiatorIyobor, Hugah Stephen, E-Agriculture Management System (a Case Study of Aflao Ketu South Municipality in Ghana). *Journal of Software Engineering and Simulation.* 6(1): p. 38-49. (2020).
47. R. C. Martin, A*gile Software Development: Principles, Patterns, and Practices*. Prentice Hall. (2002).
48. B. Rumpe and A. Schröder, Quantitative Survey on Extreme Programming Projects. *arXiv preprint arXiv:1409.6599.* (2014).
49. F. Anwer, S. Aftab, S. M. Shah, and U. Waheed, Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum. *International Journal of Computer Science Telecommunications.* 8(2): p. 1-7. (2017).
50. P. Kruchten, S. Fraser, and F. Coallier, *Agile Processes in Software Engineering and Extreme Programming.* Springer.(2019).
51. J. Shore, *The Art of Agile Development: Pragmatic Guide to Agile Software Development.* " O'Reilly Media, Inc.".(2007).
52. D. Mancl and S. D. Fraser. Xp 2019 Panel: *Agile Manifesto–Impacts on Culture, Education, and Software Practices. in International Conference on Agile Software Development*. (2019).
53. S. Mohammadi, B. Nikkhahan, and S. Sohrabi, Challenges of User Involvement in Extreme Programming Projects. I*nternational Journal of Software Engineering Its Applications*. 3(1): p. 19-32. (2009).
54. A. Firdaus, I. Ghani, and S. R. Jeong, Secure Feature Driven Development (Sfdd) Model for Secure Software Development. *Procedia-Social and Behavioral Sciences.* 129: p. 546-553. (2014).
55. A. F. Arbain, I. Ghani, and S. R. Jeong, A Systematic Literature Review on Secure Software Development Using Feature Driven Development (Fdd) Agile Model. *Journal of Internet Computing and services.* 15(1): p. 13-27. (2014).
56. K. Pathak and A. Saha, Review of Agile Software Development Methodologies. *International Journal Of Advanced Computer Science And Applications.* 3(2). (2013).
57. P. Aggarwal and R. M. Chandani, Agile Methodology Influence on Sdlc (Software Development Life Cycle). *Studies in Indian Place Names.* 40(50): p.

4579-4589. (2020).

58. S. A. K. Gahyyur, A. Razzaq, S. Z. Hasan, S. Ahmed, and R. Ullah, Evaluation for Feature Driven Development Paradigm in Context of Architecture Design Augmentation and Perspective Implications. *International Journal Of Advanced Computer Science Applications*. 9(3): p. 236-247. (2018).
59. C. Budoya, M. Kissaka, and J. Mtebe, Instructional Design Enabled Agile Method Using Addie Model and Feature Driven Development Method. *International Journal of Education Development using ICT.* 15(1). (2019).
60. M. M. Jha, R. M. F. Vilardell, and J. Narayan. *Scaling Agile Scrum Software Development: Providing Agility and Quality to Platform Development by Reducing Time to Market. in Global Software Engineering (ICGSE), 2016 IEEE 11th International Conference on. (*2016).
61. D. P. Harvie and A. Agah, Targeted Scrum: Applying Mission Command to Agile Software Development. *IEEE Transactions on Software Engineering.* 42(5): p. 476-489. (2016).
62. M. Girma, N. M. Garcia, and M. Kifle. *Agile Scrum Scaling Practices for Large Scale Software Development. in 2019 4th International Conference on Information Systems Engineering (ICISE).* (2019).
63. T. Dingsøyr, T. Dybå, M. Gjertsen, A. O. Jacobsen, T.-E. Mathisen, J. O. Nordfjord, K. Røe, and K. Strand, Key Lessons from Tailoring Agile Methods for Large-Scale Software Development. *IT Professional.* 21(1): p. 34-41. (2019).
64. M. B. Firdaus, I. M. Patulak, A. Tejawati, A. Bryantama, G. M. Putra, and H. S. Pakpahan. *Agile-Scrum Software Development Monitoring System. in 2019 International Conference on Electrical, Electronics and Information Engineering (ICEEIE).* (2019).
65. R. Vallon, B. J. da Silva Estacio, R. Prikladnicki, T. J. I. Grechenig, and S. Technology, Systematic Literature Review on Agile Practices in Global Software Development. *Information Software Technology.* 96: p. 161-180. (2018).
66. R. Banfield, C. T. Lombardo, and T. Wax, D*esign Sprint: A Practical Guidebook for Building Great Digital Products.* " O'Reilly Media, Inc.".(2015).
67. L. R. Vijayasarathy and C. W. Butler, Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter? *IEEE Software.* 33(5): p. 86-94. (2016).
68. F. S. Silva, F. S. F. Soares, A. L. Peres, I. M. de Azevedo, A. P. L. Vasconcelos, F. K. Kamei, and S. R. de Lemos Meira, Using Cmmi Together with Agile Software Development: A Systematic Review. *Information and Software Technology.* 58: p. 20-43. (2015).
69. P. Lata, Agile Software Development Methods. *International Journal of Computer Science.* 20. (2016).
70. L. Rusdiana, Dynamic Systems Development Method Dalam Membangun Aplikasi Data Kependudukan Pada Kelurahan Rantau Pulut. *Jurnal Transformatika.* 16(1): p. 84-90. (2018).
71. A. Cockburn, Writing Effective Use Cases, the Crystal Collection for Software Professionals, 2000. Addison-Wesley, MA, USA, http://www.amazon.com/Writing-Effective-Cases ….
72. D. Turk, R. France, and B. Rumpe, Assumptions Underlying Agile Software Development Processes. *arXiv preprint arXiv:*1409.6610. (2014).
73. E. Kupiainen, M. V. Mäntylä, and J. Itkonen, Using Metrics in Agile and Lean Software Development–a Systematic Literature Review of Industrial Studies. *Information and Software Technology.* 62: p. 143-163. (2015).
74. J. Tripp and A. Aitken. *Introduction to Agile and Lean Software Engineering Minitrack. in System Sciences (HICSS), 2015 48th Hawaii International Conference on.* (2015).
75. T. Stone, M. Gershon, and H. Meyers, Investments in Lean Agile Software Development Training: The Impact on Productivity and Financial Performance. *Available at SSRN* 3359015. (2019).
76. P. Rodríguez, M. Mäntylä, M. Oivo, L. E. Lwakatare, P. Seppänen, and P. Kuvaja, Advances in Using Agile and Lean Processes for Software Development, in *Advances in Computers. Elsevier.* p. 135-224.(2019).
77. K. C. Kodali. *Development of Web Based Application for Supply Chain Management. in n Proceedings of the 19th Panhellenic Conference on Informatics*. (2015). (2016 of Conference).
78. D. Karagiannis. *Agile Modeling Method Engineering. in Proceedings of the 19th Panhellenic Conference on Informatics.* (2015).
79. N. Santos, J. Pereira, F. Morais, J. Barros, N. Ferreira, and R. J. Machado, An Agile Modeling Oriented Process for Logical Architecture Design, in *Enterprise, Business-Process and Information Systems Modeling.* Springer. p. 260-275.(2018).
80. M. Amir, K. Khan, A. Khan, and M. Khan, An Appraisal of Agile Software Development Process.

International Journal of Advanced Science & Technology. 58(56): p. 20. (2013).

81. N. Santos, J. Pereira, F. Morais, J. Barros, N. Ferreira, and R. J. Machado. Incremental Architectural Requirements for Agile Modeling: *A Case Study within a Scrum Project. in Proceedings of the 19th International Conference on Agile Software Development: Companion.* (2018).
82. Sukasi, Agile Modeling (Am) - Product Development. (2017).
83. F. Mognon and P. C. Stadzisz. Modeling in Agile Software Development: *A Systematic Literature Review. in Brazilian Workshop on Agile Methods.* (2016).
84. J. Highsmith, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems.* Addison-Wesley.(2013).
85. A. Agovic and A. Agovic, Universal and Adaptive Software Development Platform for Data-Driven Applications. 2016, Google Patents.
86. A. F. Chowdhury and M. N. Huda. Comparison between Adaptive Software Development and Feature Driven Development. in *Proceedings of 2011 International Conference on Computer Science and Network Technology.* (2011).
87. Kanban and Adaptive Software Development. (2013).
88. N.-T. Huynh, M.-T. Segarra, and A. Beugnard. A Development Process Based on Variability Modeling for Building Adaptive Software Architectures. in *2016 Federated Conference on Computer Science and Information Systems (FedCSIS).* (2016).
89. H. Lei, F. Ganjeizadeh, P. K. Jayachandran, and P. Ozcan, *A Statistical Analysis of the Effects of Scrum and Kanban on Software Development Projects. Robotics and Computer-Integrated Manufacturing.* 43: p. 59-67. (2017).
90. I. Shamshurin and J. S. Saltz, Using a Coach to Improve Team Performance When the Team Uses a Kanban Process Methodology. *Governance, governmentality project performance: the role of sovereignty.* 7(2): p. 61-77. (2019).
91. J. Saltz and R. Heckman, Exploring Which Agile Principles Students Internalize When Using a Kanban Process Methodology. *Journal of Information Systems Education.* 31(1): p. 51. (2020).
92. K. Bhavsar, V. Shah, and S. Gopalan, Scrumbanfall: An Agile Integration of Scrum and Kanban with Waterfall in Software Engineering. *International Journal of Innovative Technology Exploring Engineering.* 9(4): p. 2075-2084. (2020).
93. Slideteam, Agile Kanban. *Romanian Journal of Information Technology and Automatic Control.* 29(4): p. 7-16. (2019).