RUJCST
جامعة الرازي
Al-Razi University
كلية الحاسوب وتقنية المعلومات
مجلة جامعة الرازي لعلوم الحاسوب وتقنية المعلومات
Al-Razi University Journal of Computer Science and Technology
علمية محكمة تصدر عن كلية الحاسوب وتقنية المعلومات - جامعة الرازي

A systematic study on Traditional software development models and Agile Software Development Methodologies

Yahya Al-Ashmoery
Department of Information Technology
Al-Razi University
Department of Mathematics & Computer
Faculty of Science
Sana'a University, Yemen
Yah.AlAshmoery@su.edu.ye

Najran Nasser
Department of Mathematics & Computer
Sana'a University
Faculty of Science
Sana'a, Yemen
Meetnajran@gmail.com

Youness Chaabi
CEISIC, Royal Institute for Amazigh
Culture, Rabat, Morocco
chaabi@ircam.ma

Hisham Haider
Department of Computer Science
Amran University
Al-Razi University
Sana'a, Yemen
hesham_haider@yahoo.com

Khaled Alwesabi
Department of Information Technology
Al-Razi University
Sana'a, Yemen
koalwesabi5022@gmail.com

Adnan Haider
Department of Computer Science
Amran University
Al-Razi University, Yemen
eng.adnanhaider@gmail.com

***Abstract*— Agile software development is one of the effective approaches that software engineering has developed to get to the final software product. Software engineering is a discipline that has undergone many improvements that aims to keep up with the new advancements in technology and the modern business requirements. For a very long time, traditional software development models like Waterfall, RAD, V-Model, and Spiral Model dominated the software industry. However, to keep up with expanding needs and technological advancements, software developers tried to investigate more advanced software development models, which eventually led to the creation of agile development models. Software development is a crucial undertaking that needs a thorough and organized manual in the form of a model of the software development process. In order to create software of the highest caliber, a good software development process model can be really helpful. This study presents a thorough analysis of the major agile values and concepts as well as the key distinctions between agile and conventional techniques. Following that, a review of the most common agile approaches is presented, along with information on their life cycles, roles, and benefits and drawbacks. the most recent cutting-edge trends that use agile development. The article also covers some of the difficulties teams could have while putting agile software development approaches into practice, as well as the advantages of employing them. Finally, it offers some advice for businesses who are thinking about implementing agile development.**

Keywords— The software development life cycle (SDLC), Agile software development, RAD, V-Model, Spiral Model.

I. INTRODUCTION

Software engineering belongs to the engineering family since it undergoes the same analysis, design, and development procedures as other engineering specialties like electrical engineering, mechanical engineering, and civil engineering. Similar to other technical disciplines, software is likewise created to serve a specific function [1].

Software engineering and development are joint endeavors, and while tasks may be divided among several teams, they must be managed and prioritized according to certain standards. Although tasks can be completed in parallel, some cannot begin before they are completed. In order to produce the finest software or product at the lowest possible cost, coordination between these activities, procedures, and teams is therefore essential. [2].

Software engineering is a field that integrates computer science and engineering processes and ideas to create software systems. It entails using methodical and disciplined ways to software development, ensuring that the program is dependable, effective, and simple to maintain. The broad area of software engineering includes a variety of tasks, such as requirements gathering, design, coding, testing, and maintenance.[3].

Software engineering is the process of using engineering concepts and procedures to create software. It entails the application of methodical, quantitative, and structured approaches to software development, allowing for the

production of high-quality software that is dependable, effective, and simple to maintain [4].
Software development is a complex process that involves several stages, from planning and design to coding and testing. The software development life cycle (SDLC) is a framework that outlines the various stages of software development, including planning, design, development, testing, deployment, and maintenance [5]. There are several SDLC models, including the Waterfall model, the Agile model, and the Spiral model.
Software developers go through a process known as the software development life cycle (SDLC) to create and maintain software. It is a framework that aids in making sure software is created in an ordered and methodical manner that satisfies the needs of users and stakeholders [6].

Agile software development approaches are becoming more and more well-liked in the software development business because of their adaptability and flexibility. Agile approaches are built on iterative and incremental development, where software is created in quick bursts, allowing for constant feedback and advancement [7].

II. Background

Software engineering is a field of study that combines computer science and engineering to create software systems. It entails the application of methodical and disciplined methodologies to software development, which guarantees that the software is dependable, effective, and simple to maintain. Requirements analysis, design, coding, testing, and maintenance are just a few of the many tasks that fall under the umbrella of the large discipline of software engineering. [8].
Software systems are now used in our daily lives as a result of the remarkable advancement in technology. Software engineering can be characterized as a dynamic field that strives to create high-quality software systems using organized project processes adapted to the concepts of user-centered design. The Software Development Life Cycle (SDLC), a set of fundamental processes used in the design, development, and testing of software programs, aids in this process. The most popular SDLC models include waterfall, V-model, spiral, and Rapid Application Development (RAD), agile which are the main software development approaches [9].
Software development projects often face challenges such as changing requirements, tight deadlines, and complex technical environments. Agile methodologies offer an alternative to traditional project management approaches by emphasizing flexibility, collaboration, and continuous improvement. Agile methodologies promote iterative development, frequent feedback, and close collaboration between team members and stakeholders [10].

A software development life cycle (SDLC) is a process that software developers go through to create and maintain software. It is a framework that helps to ensure that software is developed in a systematic and organized way, meeting the needs of the users and stakeholders [11].

The SDLC comprises several stages, including requirements gathering, design, development, testing, deployment, and maintenance. Software engineering principles are applied throughout each stage of the SDLC, with a focus on ensuring that the software is efficient, reliable, and easy to maintain [12].

The Waterfall Model, V-Model, and RAD are examples of traditional software development techniques, also referred to as "heavyweight methodologies". These methods are predicated on a series of steps that must be carried out sequentially, such as specifying requirements, creating a solution, testing it, and deploying it. According to conventional software development methodologies, a trustworthy set of requirements must be created and documented at the beginning of a project. [13] .

Traditional software development methodologies, such as the waterfall model, are often characterized by their emphasis on planning and documentation. These methodologies can be effective for projects with well-defined requirements and a stable environment. However, they can be less effective for projects with changing requirements or a dynamic environment [14].

A linear approach to software development is the waterfall model. It consists of a number of successive phases, each of which must be finished before the subsequent phase may start. For projects with clearly specified criteria and a set budget and timetable, the waterfall SDLC is a solid option. [15].

According to theory, there are five stages involved in creating and implementing computer software in a software development life cycle (SDLC) model.

- ❖ Planning
- ❖ Analysis
- ❖ Design
- ❖ Implementation
- ❖ Testing, Deployment and Evolution

III. Types of Common Software development process MODELS

There are several common software development process models that organizations can use to manage the software development life cycle. Here are some of the most popular ones:

1. ***Waterfall model***:

The Waterfall model is a conventional approach to software development that has been in use since the 1970s. It is a sequential process that requires each phase to be finished before moving on to the next. This model is frequently criticized for being rigid and not accommodating changes in requirements, which can result in delays and cost overruns. However, it is still used in some industries, such as construction and manufacturing, where the requirements are clearly defined [15].
The Waterfall Model is a traditional approach to software development that is based on a linear, sequential process. It consists of several phases, such as requirements gathering, design, implementation, testing, and maintenance, and each phase must be completed before moving on to the next [16].

The Waterfall Model is often depicted as a cascade, with each phase flowing down into the next.
The requirements are gathered and recorded at the start of the Waterfall Model process, and the design is based on them. After the design is finished, the development of the software based on the design takes place during the implementation phase. After the implementation is finished, the program is tested to make sure it complies with the specifications and is free of errors. Finally, the software is deployed and maintained after testing is over.

Despite its drawbacks, the Waterfall Model is still widely used to describe the software development process, especially in fields where the requirements are well-established and unlikely to change. It gives software development an organized approach, with each phase building on the one before it, and it's frequently employed for big, complex projects that need careful planning and documentation [17].

Advantages of the Waterfall Model

- Easy to Understand:
- Individual Processing
- Clear and well-defined
- Easy to manage
- Clear Milestones:
- Document driven

Disadvantages of the Waterfall Model:

- Limited flexibility.
- Difficult to accommodate Change Requests.
- No Overlapping of Phases.
- Limited customer involvement.
- High risk.
- Late Defect Detection.
- Lengthy Development Cycle,

The Waterfall Model illustrated in figure 1 below.

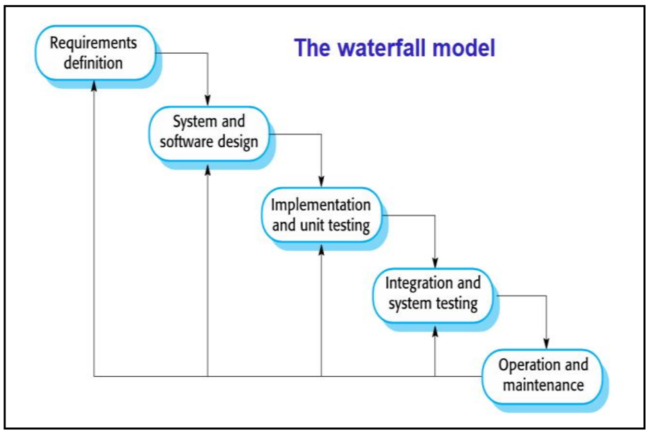
The waterfall model
Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Fig. 1: The Waterfall Model Process

2. V-Model:

The V-model is a software development model that is used to describe the different stages of the software development life cycle (SDLC). It is called the V-model because the process flow is represented by a V-shape, which shows the relationship between each stage of the SDLC and its corresponding testing phase [17].

This model resembles waterfall model in its sequential path execution of processes where each stage is finalized before proceeding to the next step [18]. Unlike waterfall method, the testing stage is emphasized in the v-model. Testing guidelines are formulated in initial stages before the implementation, and other stages are preceding coding. Prior to the development process, a system plan is designed based on the system functionality to meet all user requirements [19]. Testing of the project is planned in parallel of corresponding to phase in V-model as mentioned in figure 2.

- ***The advantages of the V-model include:***
  - Easy to understand and apply.
  - Testing-related tasks like planning and developing tests take place before coding. This saves a ton of time. higher likelihood of success than the waterfall model, therefore.
  - Proactive defect monitoring, which refers to the early discovery of faults.
  - Prevents the downward flow of faults.
  - Effective for small tasks with clear criteria.
  - ***Process that is explicit and well-defined:*** It is easier to monitor and supervise the development process since the V-model for software development offers a process that is distinct and well-defined. As a result, there may be an increase in the development team's efficacy and efficiency.
  - ***Risk reduction:*** The V-model can assist in reducing overall project failure risk by identifying and resolving problems early in the development phase.

- ***The V-model has some drawbacks:***
  - ***Lack of flexibility:*** The V-model is a rigid and inflexible paradigm that may not be suitable for all software development endeavors. Making adjustments to requirements or design modifications as the project progresses could be difficult.
  - ***High cost: The*** V-model requires a lot of preparation and documentation, which could be expensive and time-consuming. This could be a problem for projects with little funding.
  - ***Limited stakeholder involvement:*** The V-model places a strong emphasis on testing and quality assurance, which may keep some stakeholders out of the development process. As a result, it's possible that stakeholders weren't properly consulted, which could lead to a software solution that didn't meet their needs.
  - ***no early software prototypes:*** There are no early software prototypes created because software is built during the implementation phase.
  - The test documents and requirement documents must be updated if any modifications are made in the middle of the project.

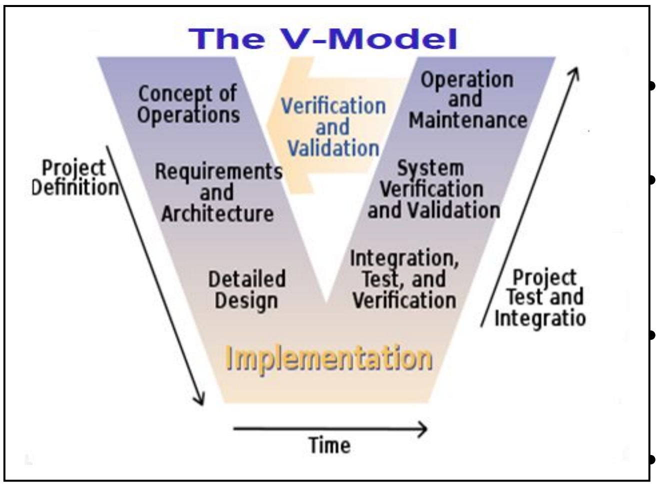
The V-Model
Concept of Operations
Verification and Validation
Operation and Maintenance
Project Definition
Requirements and Architecture
System Verification and Validation
Detailed Design
Integration, Test, and Verification
Project Test and Integratio
Implementation
Time

Fig. 2: The V-Model

3. *Spiral Model:*

The spiral model is a software development model that combines elements of both the iterative and incremental development models, as well as the Waterfall model. It is called the spiral model because the process flow is represented by a spiral that shows the iteration and repetition of the development process [20].

Spiral model: The Spiral model is a risk-driven approach to software development that emphasizes risk analysis and management. It consists of several cycles, each of which includes planning, risk analysis, development, and testing. Each cycle builds upon the previous one, with feedback and changes incorporated into subsequent cycles. This model is often used for complex projects with high risks and uncertainties [21].
The spiral model consists of four major phases, which are repeated in a spiral fashion:

1. ***Determine Objectives:*** Each cycle in the spiral begins with the determination of the cycle's objective, the numerous options available for accomplishing the goals, and the constraints that are present.
2. ***Risk Assessment and reduction:*** This phase sees the identification, evaluation, and prioritization of the project risks as well as the development of management and mitigation plans.
3. ***Development and Test:*** Creating strategies to address risks and uncertainties is the following phase. Benchmarking, simulation, and prototyping are a few possible steps in this process.
4. ***Planning:*** Finally, the next step is planned. The project is evaluated, and a decision is taken regarding whether to move forward with another spiral period. Plans for the project's next phase are created if it is decided to keep it. We display the Spiral Model in Figure 3.

The Spiral Model has the following advantages:

- ***Risk management:*** The spiral approach emphasizes risk management heavily, which helps with early risk detection, analysis, and mitigation. By doing this, the likelihood that a project will fail as a whole can be reduced.
- ***Flexibility:*** It is more flexible than the Waterfall model, which allows for changes and modifications to be made to the software throughout the development process.
- ***Participation of stakeholders and users***: The spiral model places a strong emphasis on involving stakeholders and users at every stage of the development process, which can help to ensure that the software satisfies their needs and requirements.
- ***Iterative testing and evaluation:*** The software can be tested and evaluated iteratively using the spiral approach, which can enhance the software's dependability and quality.
- Allows for the extensive use of ***prototypes***.
- More ***precise requirement*** capturing is possible.
- A better risk management strategy can be achieved by breaking development up into smaller pieces and developing the riskier portions sooner.

The disadvantages of the Spiral Model are as follows

- ***Complexity:*** The spiral model can be more complex and difficult to manage than other software development models because of the iterative and repeated structure of the process..
- ***Cost:*** Due to the additional level of planning, analysis, and testing needed, the spiral approach may be more expensive than other software development models.
- ***Time-consuming:*** Due to the spiral model's iterative process and requirement for ongoing testing and assessment, it may take longer than other software development methods to complete.
- End of the project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Documentation is overly required for the large number of intermediate phases.

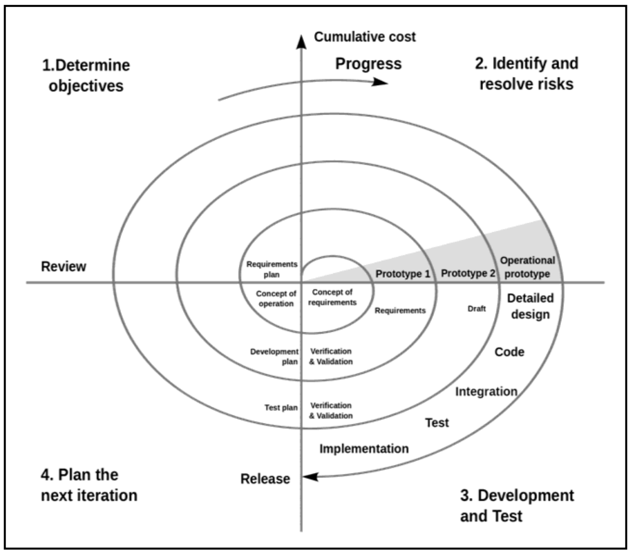

Fig. 3: The Spiral Model

4. *Rapid Application Development Model:*

Rapid application development is a method of software development that places an emphasis on rapid prototyping over in-depth planning. A prototype is a working model that is functionally equivalent to a product component [22]. The RAD model allows for quicker product delivery because functional modules are simultaneously developed as prototypes and then assembled to build the whole product. Because there isn't any considerable preplanning, incorporating the changes during the development process is easier. Small teams of developers, domain experts, client representatives, and other IT resources work iteratively on their component or prototype in RAD projects. These initiatives likewise use an incremental and iterative approach. For this technique to be successful, it is essential to ensure that the produced prototypes are reusable [23].

The Rapid Application Development (RAD) model is a software development methodology that emphasizes rapid prototyping and iterative development. The RAD model is designed to reduce the development time and costs associated with traditional software development methodologies [24].

In the RAD model, the development process is broken down into smaller, more manageable iterations or prototypes. Each prototype is developed quickly and tested to ensure that it meets the requirements of the end-users. Feedback is then gathered from the end-users, and any necessary changes or improvements are made to the prototype before moving on to the next iteration [25].

The RAD model is often used in projects where time-to-market is critical. It is also useful in projects where the requirements are not well-defined or are likely to change frequently. The RAD model promotes collaboration and communication between developers and end-users, which can lead to a better understanding of the project requirements and ultimately, a better end product.
The following illustration the RAD Model in Fig 4.

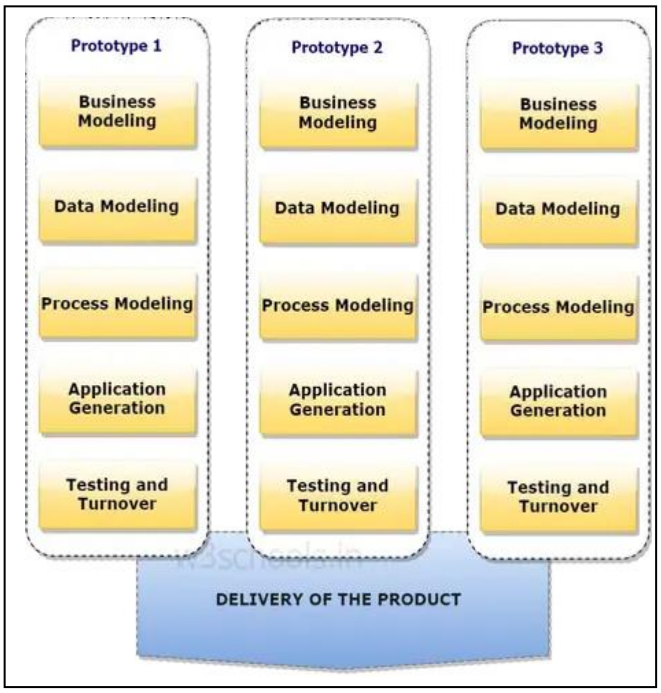

Fig. 4: The RAD Model

The advantages of the RAD Model are as follows:

- ***Speed:*** The RAD methodology allows for the rapid development of software. Since the RAD methodology places a strong emphasis on prototype and iteration, it can develop usable software more quickly than certain other techniques.
- ***Flexibility:*** Iterative procedures make it simple to integrate updates and changes as they occur, and the RAD model is flexible enough to be readily adjusted to meet new requirements.
- ***Cost-effective software development:*** The RAD model can be effective because it stresses prototype and iteration, which can ultimately lower development costs. Early issue detection helps developers avoid expensive fixes that might be required if they wait until later in the development process.
- ***Improved user participation***: By include them in the prototyping and iterative feedback cycles, the RAD model actively involves end users in the development process. This leads to a greater understanding of user needs and requirements, which ultimately results in a more satisfying finished product.
- It is possible to adapt to changing requirements.
- Progress can be measured.
- Utilizing robust RAD tools can reduce the amount of time spent iterating.
- Productivity with fewer people in a short time.
- Reduced development time.
- Increases reusability of components.
- Each phase in RAD brings highest priority functionality to the customer.

The disadvantages of the RAD Model are as follows:

- ***Quality Control:*** The RAD approach places a lot of focus on speed and flexibility, sometimes at the sacrifice of quality assurance. Software is created so quickly that there may not be enough time to thoroughly test and debug the application.
- ***Limited scalability:*** Due to its limited scalability, the RAD model is best suited for small to medium-sized applications. Larger projects could require a more structured approach to development.
- ***High dependency on individuals:*** An expert team of developers, designers, and users is necessary for the RAD paradigm. The RAD paradigm might not work if the right team is not in place.
- All application is not compatible with RAD.
- On the high technical risk, it's not suitable.
- Management complexity is more.
- Required user involvement.

5. *Agile software development Model*

The Agile software development model is an incremental and iterative method for creating software that prioritizes customer satisfaction, cooperation, and adaptability. The Agile model emphasizes the delivery of functional software in tiny, incremental releases, with each release giving end users access to additional features and capabilities [26].

Agile development involves breaking down the development process into smaller, more manageable parts called sprints, which typically last for 1-4 weeks. During each sprint, the development team works on a specific set of tasks, such as developing new features, fixing bugs, or improving existing functionality. At the end of each sprint, the team delivers a working piece of software, which can be reviewed by the product owner or stakeholders [27].

One of the key principles of Agile development is the importance of collaboration and communication between the development team and the stakeholders. This includes regular meetings, such as daily stand-ups and sprint retrospectives, to ensure that everyone is on the same page and any issues or concerns are addressed in a timely manner [28].

Another important aspect of the Agile model is the focus on adapting to change. Agile development recognizes that requirements and priorities can change over time, and therefore, the development process should be flexible enough to accommodate these changes. This is achieved through continuous feedback and iteration, with the aim of delivering the most value to the customer [29].

A. *The following steps are usually included in the agile software development process::*

1. ***Gathering of Requirements:*** The customer's requirements for the software are gathered and prioritized.
2. ***Planning:*** The development team develops a delivery plan for the program, outlining the features that will be provided in each iteration.
3. ***Development:*** The development team works to build the software, using frequent and rapid iterations.
4. ***Testing:*** The software is put through a rigorous testing process to make sure it is high-quality and fits the needs of the customer.
5. ***Deployment:*** The software is deployed and put into use.
6. ***Maintenance:*** The software is maintained to make sure that it continues to meet the demands and expectations of the customer.

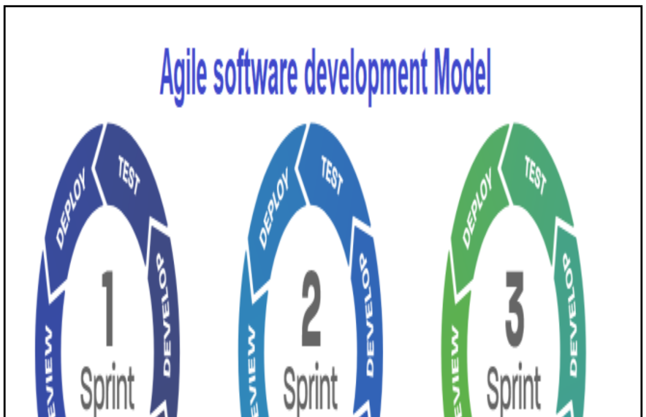

Fig. 5: The Agile software development model

The 'Agile Software Development manifesto' was published in 2001 as a consequence of a formal partnership between 17 software engineering consultants that created and supported lightweight adaptive methodologies [30]. It outlines a set of ideals and principles for software and system agility. Four ideals and twelve principles supported and made up the essence of being agile in this concept. These values and principles serve as the foundation for the software development process and provide the distinctive qualities for any approach with the agility feature [31]. According to the Manifesto, whenever a decision needs to be taken, emphasis should be placed on the matters that fall to each fundamental value's left rather than its right. The following are these four values [30]:

1. **Individuals and interactions** over processes and tools.
2. **Working software** over comprehensive documentation.
3. **Customer collaboration** over contract negotiation.
4. **Responding to change** over following a plan.

PRINCIPLES BEHIND THE AGILE MANIFESTO

We follow these principle [30]:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The advantages of the Agile Model are as follows:

- ***Flexibility:*** Agile methodologies are designed to be adaptable and flexible in response to changing requirements. Since the development process is iterative, changes can be made at any stage, which is useful in projects when the requirements are not quite apparent at the outset.
- ***Reducing risk:*** The Agile model promotes frequent testing and feedback, which can lower the likelihood of delivering a product that doesn't match customer expectations or has significant problems.
- ***Customer satisfaction:*** The Agile methodology places the interests of the client first by involving them in the development process through constant feedback and cooperation. By doing so, it is ensured that the final product meets the needs and preferences of the client.
- ***Improved teamwork:*** The Agile methodology encourages collaboration and teamwork amongst stakeholders, testers, and developers. This might result in greater communication and problem-solving skills, which would ultimately result in better work**.**
- ***Reduced time to market:*** The Agile model's focus on delivering functional software in manageable chunks may lead to a quicker time to market. This can be helpful, particularly in industries where time-to-market is crucial.

The disadvantages of the Agile Model are as follows:

- ***Lack of predictability:*** Because of the Agile model's high flexibility and adaptability, it may be difficult to foresee how a project will ultimately turn out. Budgeting, scheduling, and other resource planning can be challenging for individuals concerned.
- ***Complexity:*** The Agile methodology may be more challenging than conventional techniques, especially for teams who are new to Agile. It requires a full understanding of Agile principles and practices, which can take time and be challenging to implement.
- ***Dependence on team members:*** The success of the Agile methodology greatly depends on the skills and commitment of each team member. If team members are not fully committed or lack the necessary skills, it could be challenging to achieve the desired results.
- ***High level of participation:*** The Agile approach requires high participation from every team member, including developers, testers, and stakeholders. Teams that are geographically dispersed or organizations with limited resources may find this challenging.
- ***Strict delivery management*** determines the scope, functionality to be delivered, and adjustments to meet deadlines.
- Depends heavily on ***customer interaction***, so if customer is not clear, team can be driven in the wrong direction.
- ***Technology transfer*** to new team members may be challenging due to a lack of documentation.

Essentially, the Agile model is a collective iteration grouped to do the development practice of a product. These procedures share some essential qualities but do have assured little differences between each of them [32]. Here are lists of different types of Agile models that are used by software development companies in Fig 5. These are:

- Scrum
- Crystal
- Kanban
- Dynamic Software Development Method (DSDM)
- Feature Driven Development (FDD)
- Extreme programming (XP)
- Lean Software Development

Amongst them, Scrum, Lean and Extreme programming are some of the most popular forms of Agile development methodologies

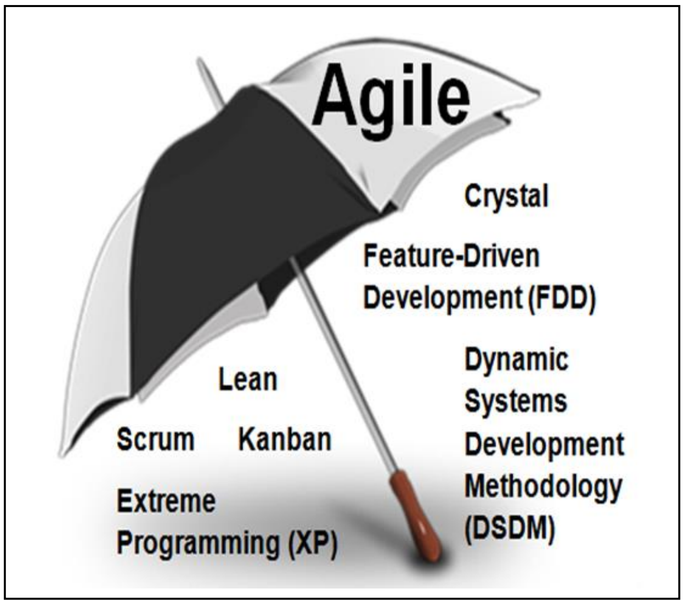
Agile
Crystal
Feature-Driven Development (FDD)
Dynamic Systems Development Methodology (DSDM)
Lean
Scrum
Kanban
Extreme Programming (XP)

Fig. 6: The different types of Agile software development models

IV. The comparison between Agile and traditional approaches to software development:

In this section, we will show a comparison between Agile and traditional approaches to software development [33][34]:

1. ***Approach:*** Agile is an iterative and incremental approach that stresses providing usable software gradually and constantly, whereas traditional SDLC follows a linear

approach that requires finishing each phase of the development cycle before moving on to the next..

2. ***Requirements:*** Agile focuses on delivering working software incrementally based on customer feedback and changing requirements. Agile teams aim to deliver the minimum viable product (MVP) early and continuously improve it based on feedback. In contrast, traditional SDLC requires a detailed and well-defined set of requirements before development begins. Traditional teams aim to deliver a fully polished and complete product at the end of the development cycle.

3. ***Planning***: Agile emphasizes flexibility and adaptability, allowing for changes to be made at any point during the development cycle. Agile teams use short planning cycles and prioritize delivering high-value features early. In contrast, traditional SDLC requires extensive planning and documentation before development begins. Traditional teams use long planning cycles and aim to deliver a complete product at the end of the development cycle.

4. ***Timeframe***: Agile aims to deliver working software incrementally and continuously, with a focus on delivering value early and often. Agile teams divide their development cycle into sprints, typically lasting two to four weeks, and aim to deliver a working product incrementally at the end of each sprint. In contrast, traditional SDLC projects typically have longer development cycles and a fixed release date. Traditional teams aim to deliver a fully polished and complete product at the end of the development cycle.

5. ***Quality***: Agile prioritizes delivering working software that meets the minimum viable product (MVP) requirements and can be improved iteratively with customer feedback. Agile teams aim to deliver value early and often, even if the product is not fully polished. In contrast, traditional SDLC focuses on delivering a highly polished final product that meets all the requirements. Traditional teams aim to deliver a complete and polished product at the end of the development cycle.

6. ***Team Structure:*** Agile promotes cross-functional teams that work collaboratively throughout the entire development cycle. Agile teams are self-organizing, with team members from different disciplines collaborating closely. In contrast, traditional SDLC often involves large teams with distinct roles and responsibilities. Traditional teams may include project managers, business analysts, developers, and testers.

7. ***Communication:*** Agile emphasizes the importance of face-to-face communication and collaboration between team members and stakeholders. Agile teams use short planning cycles, daily stand-up meetings, and regular retrospectives to encourage collaboration and communication. In contrast, traditional SDLC relies heavily on documentation and formal communication channels. Traditional teams use long planning cycles, formal documentation, and meetings to communicate progress and issues.

8. ***Risk management:*** Agile has a more proactive approach to risk management, with risks identified and addressed throughout the development cycle. Agile teams use short planning cycles, user stories, and daily stand-up meetings to identify and address risks early. In contrast, traditional SDLC typically has a more reactive approach to risk management, with risks identified and addressed during specific phases of the development cycle. Traditional teams use formal documentation and meetings to identify and address risks.

V. Scrum Software Development Framework

Popular Agile software development framework Scrum has a focus on adaptability, teamwork, and continual improvement. Scrum is a well-liked option for teams looking to adopt Agile software development processes since it is a simple, lightweight framework. [37].

In the broader Agile technique, which has dominated software development for the past 20 years, the Scrum framework is the most well-liked approach. Although accurate figures are difficult to find due to the market's extreme fragmentation, the vast majority of software development teams utilize a project management methodology that may be loosely categorized as Agile [38].

According to the Scrum Guide [39], “Scrum is a framework using within which people can address complex adaptive problems, while productively and creatively delivering products of highest value.”

While the concept borrowed its name from a technique in the Rugby game, it was Ken Schwaber and Jeff Sutherland who created the agile scrum methodology in 1995. Eventually, the concept developed into a cohesive set of practices, and with the launch of the Scrum Alliance, there are over one million Scrum-certified developers at present.

In practice, Scrum is designed to be lightweight and easy to understand. We break projects into pieces (sprints) that allow greater transparency and numerous opportunities to inspect and adapt the next sprint, the overall project, and the process.

Scrum is inherently customer-centric, recognizing that requirements often change and the overall product backlog will continue to evolve after each sprint cycle. Using a process designed with these realities in mind, we can complete the end product faster and ensure it better meets the client’s goals.

According to the 15th State of Agile Report [40], 94% of the 1382 international respondents' employers had adopted Agile. 66% of those who claimed they used a Scrum framework to implement an Agile methodology variation at the team level were within the majority.

Core Concepts of Scrum:

1) Scrum Artifacts

There are three artifacts in Scrum: the product backlog, the sprint backlog, and the product increment:

- **Product Backlog:** An ordered list of everything a team needs to complete to build the product. Due to the nature of Scrum, the list is constantly evolving with new feedback and ideas.

- **Sprint Backlog:** A subset of the product catalog, i.e., a fixed set of tasks your team needs to complete in a single development cycle (sprint).
- **Product increment:** We achieve product increment after completing the sprint backlog. The “completion” is pre-defined, including testing and/or complete approval.

2) Scrum Roles

The average scrum team consists of 4-8 people falling into one of three roles:

- **Product owner:** A product owner owns the product’s vision and decides on the product backlog. They also relay between the team and various stakeholders/customers.
- **Scrum Master:** A Scrum master’s role is to support the team by removing any obstacles to success. The Scrum Master is adept at facilitating team communication in daily scrum meetings, encouraging process improvements, and ensuring the team adheres to Scrum principles. This duality of roles (supporting and leading the team) is also called a “servant-leader” role.
- **The Development Team:** The scrum development team consists of developers, UX/UI designers, and QA team members who contribute to each sprint. As needs change over time, the composition of the team is self-organizing.

3) Scrum Events

The Scrum framework is based on incremental products developed in time-boxed events (e.g., sprints of equal duration). The average sprint event length is 2-4 weeks, with an average of 5 sprint events per project.

Each Scrum event includes the following five components:

- Sprint Planning Meeting
- The Sprint
- Daily Scrum Meetings
- Sprint Review
- Sprint Retrospective

4) Scrum Core Values

The scrum development approach has five core values and six principles and principles that all team members share to help them work together as a cohesive unit.

- **Courage:** The team must feel confident to share ideas and feedback, speak up about obstacles, and push back when needed.
- **Focus:** The time-boxed design of each sprint helps scrums stay laser-focused on a finite amount of work. We reinforce focus in Scrum via daily updates on ongoing tasks.
- **Commitment:** A scrum team is like any relationship: open communication, trust, and communication are essential to agile teams working well together and committing to tasks they believe they can complete.
- **Respect:** Cross-functional teams often come with competing input, but with that must come respect for ideas and individuals.
- **Openness:** Hand in hand with courage, your team members must be open to the feedback received from others and be willing to adapt to feedback from the team, stakeholders, or customers.

Advantage of using Scrum framework:

1. ***Improved Productivity:*** Teams can operate more effectively and efficiently thanks to Scrum. Scrum aids teams in setting priorities and maintaining concentration on the most crucial tasks at hand by focusing on producing working software in brief iterations..

2. ***Increased Transparency:*** Scrum offers both team members and stakeholders a high level of transparency. Regular gatherings, such as the Daily Scrum, Sprint Review, and Sprint Retrospective, make it easier for everyone to monitor progress, spot problems, and make the required corrections.
3. ***Flexibility and Adaptability:*** Scrum is made to be agile and responsive to changing conditions. The framework enables teams to swiftly modify their strategies and goals in response to fresh data or market shifts..

4. ***Better Communication and Collaboration:*** Scrum encourages team members to collaborate and communicate often. The framework promotes honest and open communication, which aids in speedy conflict resolution and enhances team dynamics.

5. ***Continuous Improvement:*** Scrum promotes continuous progress by requiring frequent review and modification. A crucial component of the framework is the Sprint Retrospective, which enables teams to evaluate their performance and find methods to streamline their procedures..

Disadvantage of using Scrum framework:

While Scrum has many advantages, there are also some potential disadvantages to consider, including:

1. ***Complexity:*** Scrum can be challenging, especially for teams who are unfamiliar with the methodology. It demands a substantial time and money commitment to implement, as well as a high level of commitment and discipline from the team.

2. ***Lack of Predictability:*** Scrum is made to be fluid and adaptable, which might make it challenging to foresee how a project will turn out. For stakeholders used to more conventional project management techniques that place a higher priority on predictability, this might be difficult.

3. ***Emphasis on Soft Skills:*** Scrum places a lot of emphasis on soft skills like cooperation, communication, and problem-solving. All teams need these talents to succeed, but they can be difficult to measure or quantify, making it difficult to assess the performance of individual team members.

4. ***Potential for Scope Creep:*** Scrum promotes adaptation and flexibility, which, if teams aren't attentive, can occasionally result in scope creep. This may cause initiatives to take longer than expected or to cost more than originally estimated.

5. ***Not Suitable for All Projects:*** Scrum works well for projects that are complex, uncertain, and subject to constant change. A more conventional project management methodology can be more suitable for projects that are straightforward and well-defined.

In comparison to Scrum's numerous advantages, its drawbacks are generally small. To decide whether to use the framework for a certain project or team, it is crucial to take into account these potential limitations..

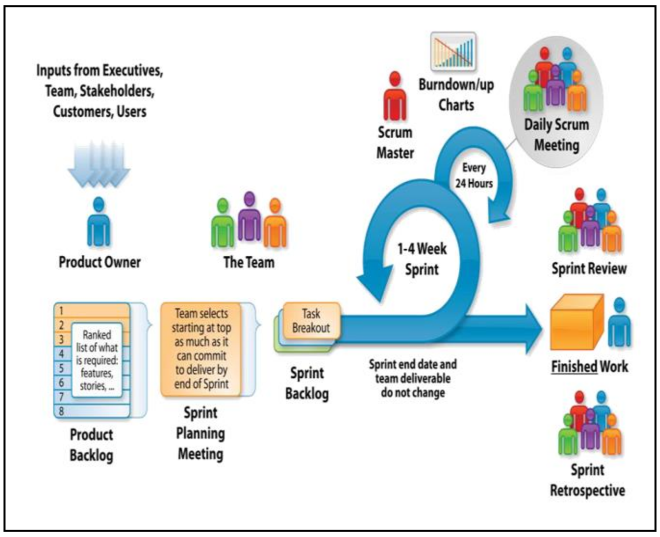
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective

Fig. 7: The Scrum Software Development Framework

VI. Important studies and statistics on the Agile Software development Methodologies

Many businesses and research firms have undertaken studies about the application of Agile development approaches. Here are some of the major conclusions from a handful of them:

1. VersionOne's State of Agile Report [41]: One of the most thorough investigations of the adoption of Agile is this yearly survey. Based on replies from more than 1,400 participants, the most recent report discovered that:
   - 98% of those surveyed had already used Agile, with Scrum being the most often used framework.
   - The most often cited advantages of adopting Agile were a better capacity to manage shifting priorities (58%), increased team productivity (54%), and improved project visibility (52%).
2. Agile Alliance's Agile 20XX survey [42]: The Agile Alliance conducts an annual survey that focuses on the adoption and usage of Agile methodologies. The most recent survey's significant conclusions include:

- 97% of respondents said their business uses Agile, with Scrum being the most widely used framework.
- Improvements in teamwork (47%) and software quality (39%) as well as shorter time to market (37%), were the main advantages of adopting Agile.

3. McKinsey's Agile at Scale survey [43]: This survey focused on large organizations' adoption of Agile at scale. Some key findings include:

- (81%) of respondents reported using Agile methodologies, with Scrum being the most popular framework.
- The most significant benefits of Agile at scale were faster time-to-market (85%), increased collaboration (82%), and higher quality software.(%81)

5. Scrum.org's State of Scrum Report [44]: This survey is focused specifically on the Scrum framework and its adoption. Key findings from the latest report include:

- (90%) of respondents reported that Scrum improved their team's ability to deliver value.
- The most common challenges with Scrum adoption were lack of experience with Agile (40%), inadequate training (36%), and difficulty changing organizational culture (34%).

Overall, these studies show that Agile methodologies are popular and offer businesses significant benefits like improved collaboration, a quicker time to market, and better software quality. Even though the Scrum framework is the most well-known, Agile methods like Kanban and Lean are rising in popularity.

Furthermore, it appears from these polls that Agile techniques are becoming more and more crucial for businesses of all sizes and in all sectors. The adoption of Agile has several advantages, including enhanced cooperation, accelerated time to market, higher-quality software, and elevated customer satisfaction. However, there are also common adoption hurdles, such as reluctance to change and a lack of expertise or training. Agile approaches can also be used in other parts of the organization to boost success and overall agility. They are not just restricted to software development.

VII. Conclusion

Agile and traditional software development approaches each have benefits and drawbacks of their own. Agile is more suited to projects that require flexibility, adaptability, and the capacity to manage changing requirements, whereas traditional SDLC is better suited to projects that have clearly defined requirements and a specified scope. The most appropriate of these two strategies will depend on the project's particular requirements and constraints.

Agile techniques encourage teamwork, continuous improvement, and the quick delivery of functional software. They are appropriate for projects with a lot of unpredictability and a need for rapid adjustments. Traditional models, on the other hand, work effectively for projects with clearly defined scopes, requirements, and deliverables.

Agile methodologies have been used by several firms in recent years to improve their software development processes. The exact requirements and goals of a project ultimately determine whether to use an Agile method or a traditional one..

Although the Agile model can be a successful software development process, there are a number of disadvantages that should be carefully considered before employing this tactic. There may be a lack of predictability, a high level of involvement, complexity, a lack of documentation, dependency on team members, and challenges in managing large teams, among others.

Choosing the software development methodology that best fits the unique needs and goals of your project is crucial in the end. The most important factor, whether you use Agile or traditional ways, is to continue to be adaptable, flexible, and open to change because software development is a discipline that is continually developing and requires ongoing learning and development..

References

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

[2] Giuffrida, R., & Dittrich, Y. (2015). "A conceptual framework to study the role of communication through social software for coordination in globally-distributed software teams".Information and Software Technology, 63, 11-30. https://doi.org/10.1016/j.infsof.2015.02.013

[3] Lalband, Neelu, and D. Kavitha. "Software engineering for smart healthcare applications." *Int J Innov Technol Explor Eng* 8.6S4 (2019): 325-331.

[4] Taskesenlioglu, Sedat, Necmettin Ozkan, and Tugba Gurgen Erdogan. "Identifying possible improvements of software development life cycle (sdlc) process of a bank by using process mining." *International Journal of Software Engineering and Knowledge Engineering* 32.04 (2022): 525-552

[5] Gurung, Gagan, Rahul Shah, and Dhiraj Prasad Jaiswal. "Software Development Life Cycle Models-A Comparative Study." *International Journal of Scientific Research in Computer Science, Engineering and Information Technology, March* (2020): 30-37.

[6] Ian Sommerville, "Software Engineering", Addison Wesley, 10th edition, 2016.

[7] Agile Business Consortium Ltd. (2017). Towards an Agile Culture. Retrieved from https://cdn.ymaws.com/www.agilebusiness.org/resource/resmgr/documents/whitepaper/towards_an_agile_culture.pdf

[8] Gurung, Gagan, Rahul Shah, and Dhiraj Prasad Jaiswal. "Software Development Life Cycle Models-A Comparative Study." International Journal of Scientific Research in Computer Science, Engineering and Information Technology, March (2020): 30-37.

[9] R. Hoda, N. Salleh, J. Grundy, H. M. Tee, Systematic literature reviews in agile software development: A tertiary study, Information and Software Technology 85 (2017) 60–70.

[10] Vijayasarathy, L. R., & Butler, C. W. (2016). "Choice of software development methodologies: Do organizational, project, and team characteristics matter?". IEEE software, 33(5), 86-94. https://doi.org/10.1109/ms.2015.26

[11] Iqbal, Syed Zaffar, and Muhammad Idrees. "Z-SDLC model: a new model for software development life cycle (SDLC)." International Journal of Engineering and Advanced Research Technology (IJEART) 3.2 (2017): 8.

[12] Kramer, Mitch. "Best practices in systems development lifecycle: An analyses based on the waterfall model." Review of Business & Finance Studies 9.1 (2018): 77-84.

[13] Gurung, Gagan, Rahul Shah, and Dhiraj Prasad Jaiswal. "Software Development Life Cycle Models-A Comparative Study." International Journal of Scientific Research in Computer Science, Engineering and Information Technology, March (2020): 30-37.

[14] Niederman, Fred, Thomas Lechler, and Yvan Petit. "A research agenda for extending agile practices in software development and additional task domains." Project Management Journal 49.6 (2018): 3-17.

[15] Kramer, Mitch. "Best practices in systems development lifecycle: An analyses based on the waterfall model." Review of Business & Finance Studies 9.1 (2018): 77-84.

[16] Aroral, Harkirat Kaur. "Waterfall Process Operations in the Fast-paced World: Project Management Exploratory Analysis." International Journal of Applied Business and Management Studies 6.1 (2021): 91-99.

[17] Kargl, Frank, et al. "A privacy-aware V-model for software development." *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019.

[18] Jovliyevich, Kholikulov Bekzod. "A Survey of Software Development Process Models in Software Engineering." *Eurasian Scientific Herald* 8 (2022): 69-72.

[19] Khan, Nabeel Asif. "Research on various software development lifecycle models." *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 3*. Springer International Publishing, 2021.

[20] Alshamrani, Adel, and Abdullah Bahattab. "A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model." *International Journal of Computer Science Issues (IJCSI)* 12.1 (2015): 106.

[21] Doshi, Dhruv, Labdhi Jain, and Kunj Gala. "Review of the spiral model and its applications." *Int. J. Eng. Appl. Sci. Technol* 5 (2021): 311-316.

[22] Egwoh, Abdullahi Yusuf, and Ogwueleka Francisca Nonyelum. "A software system development life cycle model for improved students communication and collaboration." *International Journal of Computer Science & Engineering Survey (IJCSES)* 8.4 (2017): 1-10.

[23] Akinsola, Jide ET, et al. "Comparative analysis of software development life cycle models (SDLC)." *Intelligent Algorithms in Software Engineering: Proceedings of the 9th Computer Science On-line Conference 2020, Volume 1 9*. Springer International Publishing, 2020.

[24] Boehm, Barry, et al. *The incremental commitment spiral model: Principles and practices for successful systems and software*. Addison-Wesley Professional, 2014.

[25] lbanna, A., & Sarker, S. (2016). "The risks of agile software development: Learning from Adopters". IEEE Software, 33(5), 72-79. https://doi.org/10.1109/ms.2015.150

[26] Anwer, F., Aftab, S., Waheed, U., & Muhammad, S. S. (2017)." Agile Software Development Models TDD, FDD, DSDM, and Crystal Methods: A Survey". International journal of multidisciplinary sciences and engineering", 8(2), 1-10.

[27] Chopade, M. R. M., & Dhavase, N. S. (2017). "Agile software development: Positive and negative user stories". 2nd International Conference for Convergence in Technology (I2CT) (pp. 297-299). IEEE. https://doi.org/10.1109/i2ct.2017.8226139

[28] Morandini, Marcelo, et al. "Considerations about the efficiency and sufficiency of the utilization of the Scrum methodology: A survey for analyzing results for development teams." *Computer Science Review* 39 (2021): 100314.

[29] Edison, Henry, Xiaofeng Wang, and Kieran Conboy. "Comparing methods for large-scale agile software development: A systematic literature review." *IEEE Transactions on Software Engineering* 48.8 (2021): 2709-2731.

[30] Beck, Kent, et al. "Manifesto for agile software development." (2001).

[31] Al-Saqqa, Samar, Samer Sawalha, and Hiba AbdelNabi. "Agile software development: Methodologies and trends." International Journal of Interactive Mobile Technologies 14.11 (2020).

[32] Jovanović, Miloš, et al. "Agile transition and adoption frameworks, issues and factors: a systematic mapping." *IEEE Access* 8 (2020): 15711-15735.

[33] Gaborov, Maja, et al. "Comparative analysis of agile and traditional methodologies in IT project management." *Journal of Applied Technical and Educational Sciences* 11.4 (2021): 1-24.

[34] Shaikh, Sarang, and Sindhu Abro. "Comparison of traditional & agile software development methodology: A short survey." *International Journal of Software Engineering and Computer Systems* 5.2 (2019): 1-14.

[35] Martin, R.C., Agile Software Development, Principles, Patterns and Practice. Prentice Hall. 2002.

[36] CollabNet VersionOne. (2019). 13th Annual State of Agile Report. CollabNet, Inc. Retrieved from https://explore.versionone.com/state-of-agile/13th-annual-state-of-agile-report/

[37] Carneiro, Laura B., Ana Carolina CLM Silva, and Luciana Hazin Alencar. "Scrum agile project management methodology application for workflow management: a case study." 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, 2018.

[38] Zayat, Wael, and Ozlem Senvar. "Framework study for agile software development via scrum and Kanban." International journal of innovation and technology management 17.04 (2020): 2030002.

[39] Schwaber, Ken, and Jeff Sutherland. "The scrum guide." *Scrum Alliance* 21.1 (2011): 1-38.

[40] I. Digital.ai Software, "15th State of Agile Report," Digital.ai, pp. 1– 23, 2021, [Online]. Available: https://stateofagile.com/#.

[41] VersionOne Inc. (2020, May 28). 14th Annual State of Agile Report. Retrieved from stateofagile.com: https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494

[42] Thonet, Claudia. "Agile Culture Change in Sales." The Agile Sales: Successfully shaping transformation in sales and service. Wiesbaden: Springer Fachmedien Wiesbaden, 2023. 77-109.

[43] Naslund, Dag, and Rahul Kale. "Is agile the latest management fad? A review of success factors of agile

transformations." *International Journal of Quality and Service Sciences* 12.4 (2020): 489-504.

[44] Scrum.org's State of Scrum Report (2019) https://www.scrum.org/resources/2019-scrum-master-trends-report