Agile Software Development

Alina-Mădălina GHEORGHE, Ileana Daniela GHEORGHE, Ioana Laura IATAN
Bucharest University of Economic Studies
gheorghealina15@stud.ase.ro, gheorgheileana15@stud.ase.ro, iatanioana15@stud.ase.ro

Agile software development represents a major departure from traditional, plan-based approaches to software engineering. The selections of appropriate software development methodologies for a given project and tailoring the methodologies to a specific requirement have been a challenge since the establishment of software development as a discipline. The research speaks about what is currently known about the benefits and limitations of agile methods and methodologies. Agile software development is a particularly important topic in software engineering and information systems. We approach the idea of the four core of agile software development that are essential to ease the process of software development.
Keywords: *software development, agile, manifesto, principles, methodologies, project life cycle, concept*
DOI: 10.24818/issn14531305/24.2.2020.08

1. Introduction

Agile is more of a philosophy, a way of approaching software development. If we were to give a definition, we would say that Agile is a set of values and principles and involves the delivery of a good software product to the customer, using an adaptive, incremental and iterative way of working by cross-functional and self-organized teams.

Agile Software development is a type of development methodology that was created due to the need of flexibility. Agile focuses on delivering individual parts of the software, in contrast with other methodologies which deliver only the entire application at the end of the development process.

Agile methodologies are an alternative to traditional project management practices. They were born in the context of software development, but today they can be used to any type of project in different fields of activity. Agile methodologies have been helping many teams deal with unpredictability within a project through incremental deliveries and iterative cycles.

The main benefit of Agile Software Development is the ability to help teams evolve in a dynamic way while still focusing on delivering high product value. Also, the collaboration is very important, as teams need to work together and understand their specific roles in the process. Another important fact is that the high-quality is maintained, since testing is performed all throughout the development process, providing enough time to encounter and establish the bugs that appear in the code, and submit these issues in time for the development team to fix them.

Previous to Agile, there was the Waterfall development methodology, which worked following the principle of very strict timelines for processes, without the ability to go back to one of them (example: the testing was done only after all the development has finished), and therefore, the client would only receive a glimpse of the product at the very end of the project. However, soon enough, the Agile methodology might be replaced by another, the DevOps one, that is growing in popularity.

Nowadays, software development it is strongly influenced by agile software development on how it is organized and conducted. This term become very familiar among developers, because they used its principles and its manifestos to coordinates and plan their everyday work and the way they communicate with their customers and external stakeholders. Also, it affects the way how software development is organized in small, medium-sized or large companies[10].

In his book, Dyba Tore say that agile software development is seen as a "reaction to plan-based or traditional methods, which emphasize a rationalized, engineering-based

approach"[11] and it incorporates extensive planning, codified processes and rigorous reuse. On the other hand, agile methods approach the idea of an unpredictable world by recognizing the value relationship between people can bring to software development.
This article consists of six sections. Section I is the introduction and provides a short description of agile software development. Section II describes the project life cycle and the three most important key concepts of a project. Section III introduces the Agile Manifesto and presents some methods of this methodology. Section IV describes the 12 principles of agile methodology and provides information on research methods. Section V focuses on results of the research and Section VI concludes the paper.

2. Project and development life cycle

A project lifecycle is the series of phases passes through from its start to its completion. It provides basic framework for managing the project. This basic framework applies regardless of the specific project work involved. The phrases may be sequential, iterative or overlapping. All project can be mapped to the generic life cycle shown in the figure 1.

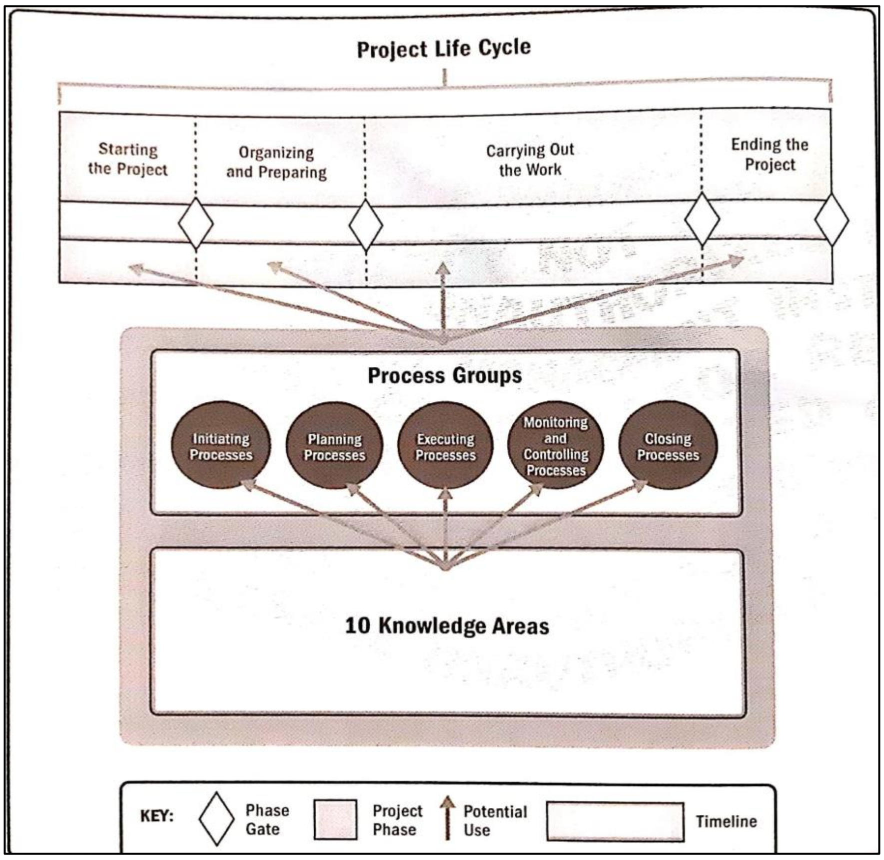


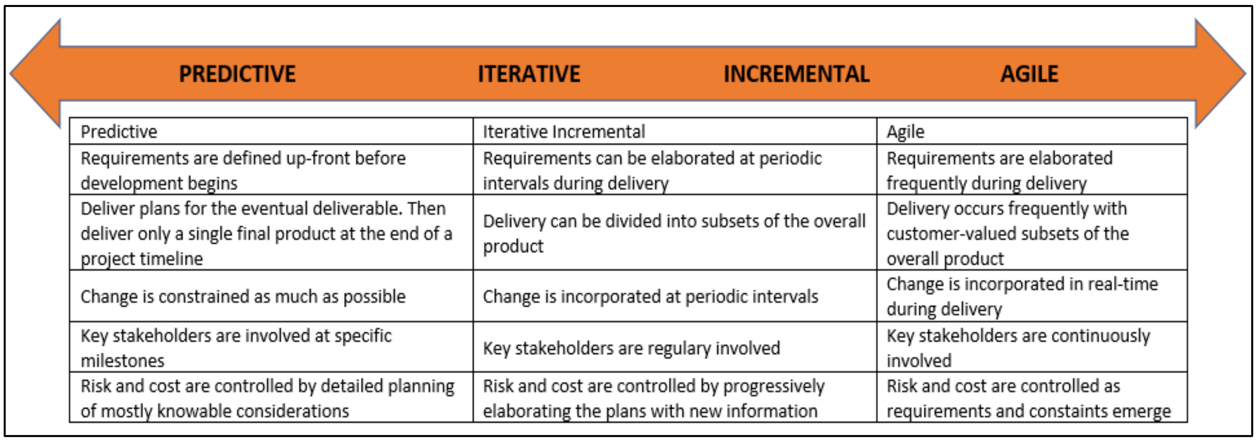
Fig. 1. Project Life Cycles [8]

Within a project lifecycle, there are general one or more phases there are associated with the development project, service result. These are called a development life cycle. Development life cycles are[8]:

1. *Predictive life cycle* where the project scope, time and scope are determined in the early phases of the life cycle. Any changes to the scope are carefully managed. Predicted life cycles may also be referred to as Waterfall lifecycle;
2. *Interactive life cycle* where the project scope is generally determined early in the project lifecycle, but time and cost estimates are routinely modified as the project team's understanding of the product increases;

3. *Incremental life cycle* where the deliverable is produced through a series of iterations that successively add functionality within a predetermined time frame. The deliverable contains the necessary and sufficient capability to be considered complete only after the final iteration;
4. *Adaptive life cycle* which are agile, iterative or incremental. The detailed scope is defined and approved before the start of an iteration. Adaptive life cycles are also referred to as agile or change-driven cycles;
5. *Hybrid life cycle* which is a combination of a predictive and an adaptive life cycle. Those elements of the project that are well known or have fixed requirements follow predicted development life cycle and those elements that are still evolving follow an adaptive development life cycle.

The continuum of project life cycle can be observed in the table below, including four out of five phases of a project life cycle.


PREDICTIVE
ITERATIVE
INCREMENTAL
AGILE

Predictive	Iterative Incremental	Agile
Requirements are defined up-front before development begins	Requirements can be elaborated at periodic intervals during delivery	Requirements are elaborated frequently during delivery
Deliver plans for the eventual deliverable. Then deliver only a single final product at the end of a project timeline	Delivery can be divided into subsets of the overall product	Delivery occurs frequently with customer-valued subsets of the overall product
Change is constrained as much as possible	Change is incorporated at periodic intervals	Change is incorporated in real-time during delivery
Key stakeholders are involved at specific milestones	Key stakeholders are regulary involved	Key stakeholders are continuously involved
Risk and cost are controlled by detailed planning of mostly knowable considerations	Risk and cost are controlled by progressively elaborating the plans with new information	Risk and cost are controlled as requirements and constaints emerge

Fig. 2. Continuum of project life cycle [8]

Forward we will detail some key concepts for project management, detailed in book "A guide to the Project Management Body of Knowledge"[8], namely :

- Key concepts for project integration management
- Key concepts for project resource management
- Key concepts for project risk management

Key concepts for project integration management include[8]:

- Project Integration Management is the specific responsibility of the project manager and it cannot be delegated or transferred. The project manager is the one that combines the results from all the other Knowledge Areas to provide an overall view of the project. The project manager is the ultimately responsible for the project as a whole.
- Projects and project management are integrative by nature, with most tasks involving more than one Knowledge Area.
- The relationships of processes within the Project Management Process Groups and between the Project Management Process

Key concepts for Project Resource Management include[8]:

- Project resources include both physical resources (equipment, materials, facilities, and infrastructure) and team resources (individuals with assigned project roles and responsibilities).
- Different skills and competences are needed to manage team resources versus physical resources.
- The project manager should be both the leader and the manager of the project team, and should invest suitable effort in acquiring, managing, motivating, and empowering team members.
- The project manager should be aware of the team influences such as the team environment, geographical location of team members, communication among stakeholders, organizational change management, internal and external

politics, cultural issues, and organizational uniqueness.

- The project manager is responsible for proactively developing team skills and competences while retaining and improving team satisfaction and motivation.
- Physical resource management is concentrated on allocating and utilizing the physical resources needed for successful completion of the project in an efficient and effective way. Failure to manage and control resources efficiently may reduce the chance of completing the project successfully.

Key concepts for Project Risk Management include[8]:

- All projects are risky. Organizations choose to take project risk in order to create value, while balancing risk and reward.
- Project Risk Management aims to identify and manage risks that are not covered by the other project management processes.
- Risk exists at two levels whiting every project: individual project risk is an uncertain event or condition that, if it occurs, has a positive or negative effect on one or more project objectives. Overall project risk is the effect of uncertainty on the project as a whole, arising from a all sources of uncertainty including individual risks, representing the exposure of stakeholders to the implications of variations in project outcome, both positive and negative.
- Individual projects risks can have a positive or negative effect on project objectives if they occur. Overall project risk can also be positive or negative.
- Risks will continue to emerge during the lifetime of the project, so Project Risk Management processes should be conducted iteratively.
- In order to manage risk effectively on a particular project, the project team needs to know what level of risk exposure is acceptable in pursuit of project objectives. This is defined by measurable risk thresholds that reflect the risk appetite of the organization and project stakeholders.

3. The state of knowledge

Agile Software Development has four core values that must be followed at all times. These values were found in 2001, by 17 professionals that gathered to discuss concepts that would help ease the process of software development, values which are the core of the Agile Manifesto.

The four core values outlined in the Agile Manifesto[1][9] are:

- *"Individual interactions are more important than processes and tools"* – this principle explains the fact that the people drive the development process and they are the ones that respond to the business needs. Therefore, the tools and processes should occupy the second place in priority, because the people can adapt way faster to the dynamic character of development.
- *"A focus on working software rather than thorough documentation"* – this value describes that in the past there were very long delays due to the fact that the programmers had to focus on documenting very thoroughly their code, and they could not focus very long on the actual implementation of their requirements. With Agile, documentation is still important, but is more succinct, usually in the form of user stories. In this way, there are less delays and the developers have a large amount of time to assign to the actual programming.
- *"Collaboration instead of contract negotiations"* – this value states that there should always be collaboration between the customer and the project manager, in order for both parties to understand the needs and capabilities of each other. This way of working is very beneficial because it provides a better understanding of the needs of the client, and therefore, the final product will meet the customer's expectations, as desired. If there would only be a negotiation at the start of the project, the client might not be aware of how the software might be built, therefore there might be dissatisfied with the result, due to the lack of communication.
- *"A focus on responding to change"* –

this principle dictates the fact that change is always welcome in the Agile process, as it is seen as a way to improve the project and provide additional value. By preforming short iterations in the Agile cycle, the developers can figure out what works and what doesn't, and change accordingly. This flexibility helps the team to modify the process to best fit their needs.

There are a few misconceptions about the Agile approach, such as the ones listed below:

- "It's unpredictable" -> it is unpredictable, but that isn't necessary a negative thing anymore, because Agile embraces it and uses it to its own advantage and produces better outputs.
- "Agile is more focused on short-term" -> on the contrary, by starting the testing earlier during development, Agile allows the programmers to make better decisions for the long-term goals indirectly.

In their research, Williams and Cockburn[12] sustain the fact that agile software development is all about feedback and continuous change and they emphasize that "software development is an empirical or nonlinear process, where short feedback-loops are necessary to achieve a desirable and predictable outcome"[10][12]. The processes in agile development are very important, they define the agility that eliminate most of the traditional methodologies of software development, promoting the rapid response to changing environments or in users requirements.

In his article[13], Conboy sustain the idea that agile software development methods contribute to one of more of the following:

- creation of change;
- pro-action in advance of change;
- reaction to change;
- learning from change.

Also, agile software development methods must contribute and should not detract from:

- perceived economy;
- perceived quality;
- perceived simplicity.

Thus, agile software development has been characterized differently than plan-based or traditional development methods, mainly with the focus adapting to change and delivering products of high quality through simple work-processes.

Nowadays, agile software development is mainly focused to adapting of change and delivery of high-quality products through simple work processes, being the main difference from traditional and plan-based software development methods. Traditional agile methods include: its fundamental assumptions, approach to control, management style, knowledge management, role assignment, role of the customer, project cycle, development model and desired organizational structure.

4. Research methods

The first step of the research is to find sources that might contain the information needed. Using books and trustful sites we can form an idea about what the concept of Agile Software Development is.

To start with, we have to follow the 12 principles of Agile to establish a proper development process. These principles[1][9][17] are:

1. *Satisfy customers through early and continuous delivery of valuable work* – Customer are happier when they receive working software at regular interval, rather than waiting long periods of time between releases.
2. *Break big work down into smaller tasks that can be completed quickly (Simplicity – the art of maximizing the amount of work not done – is essential.)*- Agile is all about doing the right amount of something at any given time, and no more. We should author user stories small enough to get the job done and no more. We should build what we know we need now. We should not build some huge framework we think we may need someday.
3. *Recognize that the best work emerges from self-organized teams* - The team knows the best way to get something done. They are the experts. However, this does not mean the right outcome will happen on its own. Each individual is at a different place in his or her personal growth and

career. The term "servant leader" has emerged in the Agile community and replaced the typical command-and-control project manager. Self-organizing teams do not happen automatically. They emerge under the proper guidance and advice of a servant leader. As a result, skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products.

4. *Provide motivated individuals with the environment and support they need and trust them to get the job done* - Motivated teams are more likely to deliver their best work than unhappy teams. Project team leaders must provide them with the motivated environment and support in order to success. The team members must trust each other and be comfortable with conflict.
5. *Create processes that promote sustainable efforts*
6. *Maintain a constant pace for completed work* - Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release. An important aspect of this is regular releases of a product.
7. *Welcome changing requirements, even late in a project* - Agile processes harness change for the customer's competitive advantage. The ability to avoid delays when a requirement or feature request changes. Agile teams work over the sub-project in certain iteration and after release, the customer may change his plan because that sub-project caused to come new ideas. Since the teams work with sub-project, it is not a big problem to add new features into product.
8. *Assemble the project team and business owners on a daily basis throughout the project* - To develop a good project, daily meetings must be organized in order to give an info about what was done since the last meeting and will be done by the next meeting. In fact, Better decisions are made when the business and technical team are aligned
9. *Have the team reflect at regular intervals on how to become more effective, then tune and adjust behavior accordingly* - Self-improvement, process improvement, advancing skills, and techniques help team members work more efficiently. People may be challenged when it comes to engaging in true self-reflection. This is all part of the Agile journey. Each of the Agile principles are interrelated. The retrospective is the perfect place for the team to reflect and improve.
10. *Measure progress by the amount of completed work* - All members of the project must remember that progress is to deliver working and satisfying project and other features are for creating Software development driven environment.
11. *Continually seek excellence* - We need to pay close attention to technical excellence and design as our product evolves. There is a balance between "Building the right thing" and "Building the thing right.". Developers must keep their code clear and tested. So, that they know at all times that their software works. The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change.
12. *Harness change for a competitive advantage*

Having these principles in mind, we can build a development process that will bring high-quality results.

We should also take into consideration the Agile Software Development cycle, which contains six steps: concept, inception, iteration/construction, release, production and retirement. We will detail each of them.

The first step, *concept*, involves the identification of the opportunities that could be pursued and establishing an estimation of the time and work that will be required to complete the project.

The second step, *inception*, identifies the programmers that will be a part of the team, the amount of money that is assigned for funding and also the initial requirements are discussed with the client.

During the third step, *iteration/construction*, the programmers start working on the code, based on the requirements and the feedback that they receive from the customer on a regular basis.
The fourth step, *release*, includes the final quality assurance tests, fixing of the remaining issues, finalization of the system and user documentation, and, eventually, the release of the final iteration into production.
During the fifth step, *production*, processes of support necessary to maintain the software are performed.
In the final step, *retirement*, all the end-of-life activities are performed, such as notifying the clients and the final migration. The system release must be removed from production.

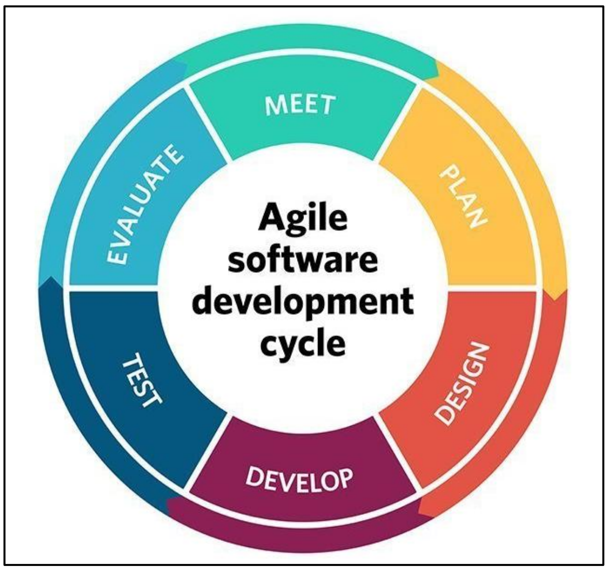


Fig. 3. The steps of Agile Software Development Cycle
Source: *https://project-management.com/10-key-principles-of-agile-software-development/*

The growth of agile is mirrored by agile research becoming a significant subdiscipline of software development in the twenty years and continuing today. In April 2018[14], a search for the keywords "agile software development" in Google Scholar for a period up to 2001 produced just over 13,000 results. The same search led to over 260,000 results for today. Also, in this time, we can identify ten principal key agile research area (which are summarized in the image below):

1. agile adoption;
2. agile methods;
3. agile practice;
4. human and social aspects;
5. agile and GSE (Global Software Engineer);
6. agile and usability;
7. agile and CMMI (Capability Maturity Model Integration);
8. organizational agility;
9. agile and embedded systems;
10. software product line engineering.

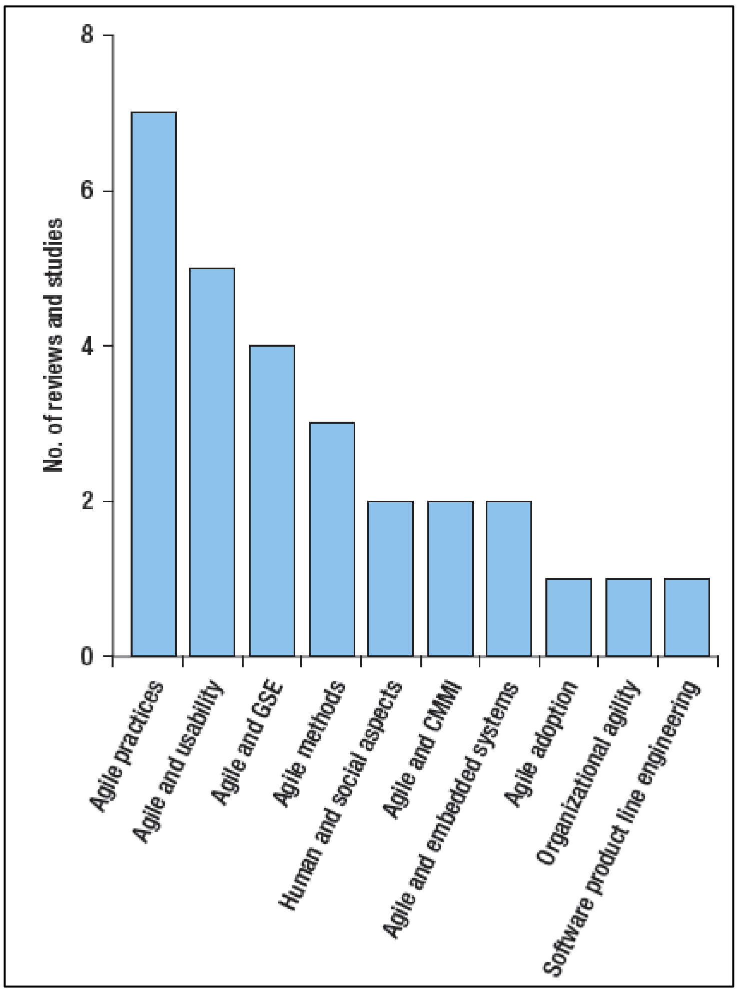


Fig. 4. Systematic literature reviews on agile topics [14]

In a review written by Cohen et. all[15], it is said that agile methods may be consolidated in the future, "just as object-oriented methods were consolidated". More than that, they believe that agile methods would "coexist" together with traditional ones, having a symbiotic relationship, where different kinds of factor, such as the number of people working on a project, innovation, will determine which process to select.

Over the time, several methods of agile software development have been proposed and used by researchers in various field. Thus, as identified in the paper "Software Development: Agile vs. Traditional"[18], some of the well-known agile methodologies, which share the same principles and values, but have different practices are:

- Development (ASD),
- Feature Driven Development (FDD)
- Crystal Clear
- Dynamic Software Development Method (DSDM)
- Rapid Application Development (RAD)
- SCRUM
- Extreme Programming (XP)
- Rational Unify Process (RUP)

5. Results

After analyzing the principles of Agile Software Development, we came to the conclusion that there are several significant benefits[6] of this methodology. Among them there are the improved product quality, the focus on business value and users, the flexibility and readiness for change, the high level of transparency and client involvement in the process. Also, we can mention the delivery of new features quickly and frequently, a cost-effective approach to product development and scaling and he predictable schedule.

We also found that there are a few key principles[3] that describes a good implementation of the Agile Software Development, namely:

- Active user involvement is imperative
- A collaborative & cooperative approach between all stakeholders is essential
- Requirements evolve, but the timescale is fixed
- Capture requirements at a high level, lightweight & visual
- The team must be empowered to make decisions
- Develop small, incremental releases and iterate
- Testing is integrated throughout the project lifecycle – test early and often
- Complete each feature before moving on to the next
- Focus on frequent delivery of products

We found that by including everyone (both the customer and the development team) in the process, the ability to get better software to the market more quickly is heightened. Therefore, collaboration and communication are the key factors that make Agile so efficient, in comparison with other development methodologies.

We also found a potential concern about Agile, namely the fact that it lacks the emphasis on technology, which could make the understanding of the concept more difficult. However, another issue could appear from the side of the sprints, namely the necessity of completing sprints on time can create a stressful work environment for software developers. This might also happen because of the need to work extra hours and stay at work late in order to meet the imposed deadlines.

The evolution of software development agile methods from traditional to what it means these days it has a particularly important role on what a company works. The mentioned methods were affected by dramatic changes in the market for software development, which lead to process adaption.

One of the important roles in agile software development projects is owned by the customer. His role works effectively and sustainable in a way that involve the whole development team. Thus, the communication between customer and development team is crucial.

6. Conclusions

Agile Software Development is a process during which the development team, the stakeholders and the end-users work together as a cross-functional team, and they identify both requirements and solutions iteratively throughout the development process.

Agile is both a philosophy and a set of guidelines, that are followed by teams in order to deliver the maximum amount of value, while minimizing the project's overhead cost. The Agile methodologies have helped many organizations to adapt quickly to the changing market conditions and also to increase client satisfaction, and improve overall efficiency.

The basic principle that the Agile methodology follows is the separation of large projects into small iterations, in such a way that, at the end of each iteration, results of high value are produced. These small iterations are called "sprints", and are designed, developed and tested individually.

Agile software development emphasizes the evolving requirements of direct user involvement in the development process, fast iterations, small and frequent releases. Improvements in the software development process include more stable requirements, earlier detection of failures, fewer testing times, increased communication and increased adaptive capacity[16]. Different methodologies require different changes in the culture of management and software development.

There are a number of factors that can directly and indirectly influence development projects in an agile environment. Adopting agile development methodologies has a positive impact on both productivity and quality. Therefore, the development team and the client are both satisfied with its implementation in software development processes.

Bibliography

[1] "Agile Software Development," [Online]. Available: https://searchsoftwarequality.techtarget.com/definition/agile-software-development. [Accessed 10 April 2020].

[2] "Agile Methodologies," [Online]. Available: https://resources.collab.net/agile-101/agile-methodologies. [Accessed 10 April 2020].

[3] "10 Key Principles of Agile Software Development," [Online]. Available: https://project-management.com/10-key-principles-of-agile-software-development/. [Accessed 10 April 2020].

[4] "Agile Software Development," [Online]. Available: https://leankit.com/learn/agile/agile-software-development/. [Accessed 10 April 2020].

[5] "Agile Software Development," [Online]. Available: https://sinovi.uk/agile-software-development. [Accessed 10 April 2020].

[6] "Agile Software Development," [Online]. Available: https://sunscrapers.com/services/agile-software-development/. [Accessed 10 April 2020].

[7] "Agile Software Development," [Online]. Available: https://www.peerbits.com/blog/agile-software-development.html. [Accessed 10 April 2020].

[8] P. M. Institute, A guide to the Project Management Body of Knowledge, GlobalStandards, 2017.

[9] P. M. Institute, Agile Practice Guide, Agile Alliance, 2017, pp. 8-9.

[10] T. Dingsøyr, B. M. Nils and D. Tore, "Agil Software Development: An Introductio and Overview," pp. 2-9, 2010.

[11] D. Tore, Improvisation in Small Software Organizations, IEEE Software, 2000, pp. 82-87.

[12] L. a. C. A. Williams, Agile Software Development: It's about Feedback, IEEE Computer, 2003, pp. 39-43.

[13] K. Conboy, Agility From First Principles: Reconstructing the Concept of Agility in Information Systems Development, Information Systems Research, 2009, pp. 329-354.

[14] R. Hoda, S. Norsaremah and G. John , "The Rise and Evolution of Agile Software Development," 2018.

[15] G. Kumar and K. B. Pradeep , "Impact of Agile Methodology on Software Development Process," pp. 46-49, 2012.

[16] P. Cohen, Advances in Computers, Advances in Software Engineering, vol. 62, 2004.

[17] Mukhamadjonov A., Khusanov Kasim, "Agile Software Development," Turin Polytechnic University, Tashkent, 2017.

[18] M. Stoica, M. Mircea, B. Ghilic-Micu, "Software Development: Agile vs. Traditional", *Informatica Economică,* vol. 17, 2013.

Ileana Daniela GHEORGHE received a Bachelor's degree in Economic Informatics from the University of Economic Studies in Bucharest in 2018 and is now studying for a master's degree in Economic Informatics at the aforementioned faculty. Her professional activity, began in September 2018 with a position of Junior Technical Consultant SQL in the company TotalSoft where she still works in a full-time position.

Alina-Mădălina GHEORGHE was rewarded with a Bachelor’s Degree in Economic Informatics, after studying at the Bucharest University of Economic Studies, Faculty of Economic Cybernetics, Statistics and Informatics. Alina is currently studying towards her Master’s Degree in the same field as her Bachelors’. She started her career as a Quality Assurance Tester at Whyttest, and as she kept learning she began applying her knowledge as a Software Developer at Suvoda Software Romania, where she is currently working.

Ioana Laura IATAN has graduated in 2018 from the Bucharest University of Economic Studies, The Faculty of Economic Cybernetics, Statistics and Informatics with a Bachelors’ in Economic Informatics and is currently doing her Masters’ Degree in Economic Informatics at the same faculty. Laura has begun her professional experience as a Junior Automation Tester for Harman Development Center Romania where she is currently employed on a full-time basis. Laura has also graduated from the Project Management course held by EXEC-EDU, a course which is accredited by the Project Management Institute, Inc.