AGILE TRANSFORMATION

In IT Organizations

For managers of IT teams, developers, and executives in the information technology industry

The most important attitude that can be formed is the desire to change.

Alisa Bobovnikova

Алиса Бобовникова
Agile Transformation in IT-organizations

INTRODUCTION

Current market conditions stimulate organizations to introduce innovative product improvement solutions. Environment changes responsiveness became an important factor in market competition. This trend is especially typical for the IT industry as this field is characterized by a lot of innovation. Transforming companies find it difficult to adapt to continuous change. This is often caused by lack of organizational flexibility and changes responsiveness practice. In order to address this issue, companies need to continuously search, evaluate, analyze and develop new tools for organizational culture.

This narrative is for those who have chosen the path of transformation to gain speed, flexibility and effectiveness. We will talk about how to improve the team's effectiveness with agile techniques and also give you an idea how to start changes and how to measure them.

Chapter 1 discusses the approach to understanding agile methodology and its main provisions. Special attention is paid to the relationship between the type of thinking and the team effectiveness. Continuous customer focus and the need for regular teams' interaction is revealed from practical examples.

Chapter 2 reflects agile management features and waterfall approach, which is also considered classic. This chapter provides these methodologies comparative analysis, highlighting distinctive features and opportunities for their use.

Chapter 3 describes agile transformation stages, most common agile frameworks, their distinctive features as well as their pros & cons.

Chapter 4 provides an overview of the main existing agile methodologies with implementation examples, as well as a step-by-step agile implementing roadmap also considering agile driven project documentation required.

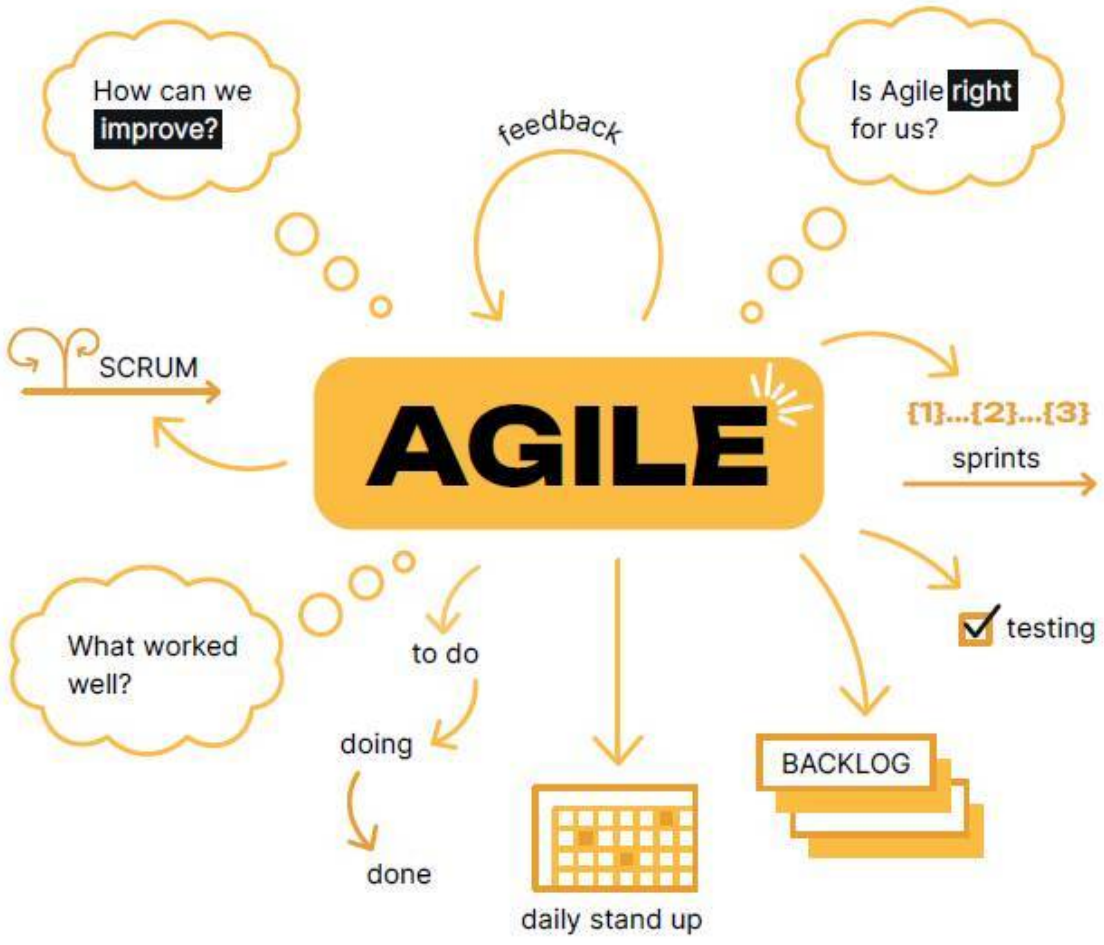
Chapter 5 and **Chapter 6** describes agile implementation effectiveness assessment methods, analyzes direct and indirect implementation process metrics as well as the technique implementation success criteria in different companies' practice, and provides key agile project's KPIs.

Chapter 7 is dedicated to the agile transformation challenges and explanation what agile coaching is, describing Agile coaching levels and tools as well as how coaching can help you with transformation.

The narrative is filled with living examples of companies implementing transformation in the IT sphere. Thanks to organizations' practice, we deliver experience that can be useful for you along this difficult path. New management practices introduction in an organization is a way that requires changes not only in actions but also in attitude. We hope you will follow this path with us.

CHAPTER 1. WHAT IS THE AGILE METHODOLOGY?

In 2021, the Agile Manifesto celebrated its 20th anniversary. The approach originated as a revolt of developers against clumsy IT corporations.



Let's figure out what Agile is, learn something from its history, and what is its difference from other software development approaches. We will also dive into some of the Agile frameworks, find out which option

is most suitable for you, and what Agile coaching is, so this knowledge will allow you to apply it in your team or company.

The material may be useful and informative for wide range of specialists dealing with software development: front-end and back-end developers, DevOps specialists and software architects as well as for team leaders, project managers, product owners, CTO's, CPO's etc.

To begin with, let's recall some facts. In 1970, **Dr. Winston Royce** issued the document that defined Waterfall[1]. Probably it was already being done by others at that time, but it is considered now to be definitive. He marked **5 principles** that must be added to reduce most of the risks of doing waterfall:

- program design comes first;
- document the design;
- do it twice;
- plan, control, and monitor testing;
- involve the customer.

Dr. Royce's last lines about waterfall were:

"[This] summarizes the five steps that I feel necessary to transform a risky development process into one that will provide the desired product. I would emphasize that each item costs some additional sum of money. If the relatively simpler process without the five complexities described here would work successfully, then of course the additional

money is not well spent. In my experience, the simpler method has never worked on large software development efforts and the costs to recover far exceeded those required to finance the five-step process listed."

In other words, Dr. Royce **suggested writing a lot of documentation, handling risks, repeating some stages several times.** The result was a heavy, in contemporary opinion, waterfall model. In an ideal world, we

would go through all levels of this system from top to bottom, like a waterfall flows, and in the end we would have a good system.

The waterfall development model quickly gained popularity in the West, and the vast majority of software development teams applied it some time ago.

Currently, Royce's name is strongly associated with the cascade methodology, while few recall that he criticized this approach, pointing out that the software development process should not resemble the conveyor line operation. IInstead of waiting for all the steps (phases) to be completed, Royce suggested using a phase-based approach with a short completion cycle of 1-2 weeks. Its essence is that initially all the requirements necessary for the project are collected, then divided into sub-projects, the target architecture is developed, the design is created, code is written in short iterations etc. The waterfall model may have

concretized and formalized the development methods existed, but it was not without its drawbacks.

Due to traditional cascade methodology imperfection, clumsiness and heaviness developers at that time already experimented with so called **“iterative-incremental” approach**[2], which gave them increased development flexibility based on regular feedback from customers and users.

The main idea of this method is to develop the system iteratively using repetitive cycles at smaller time periods (incrementally), allowing software developers to take the advantage of what was learned during the earlier parts or versions of the system development. Learning occurs both in the development and use of the system, where possible key process steps begin with the simple implementation of a subset of software requirements and iteratively improve the evolving versions until the full system is implemented. At each iteration, design changes and new features are added.

Experiments continued in the 70s, and in the 80s of the 20th century, laying the groundwork for what would become the Agile methodology. By the beginning of the 90s the iterative-incremental approach was world-famous and studied.

Now it is clear that for the Agile appearance at that time only the human dimension inquiring strive to understand human traits and how to incorporate that understanding into management planning and actions was missing, exactly teamwork and stake on people and their interaction. By the beginning of the 90s, two key aspects were trained and tested, which will form the basis of Agile: iterative-incremental approach and team work, when the project is executed by a group of people interacting with each other as efficiently as possible, which also allows to implement projects of any scale.

The 80-90s were marked by personal computers (Apple Macintosh, PC-based and others) mass distribution. Millions of new computers emerged in ordinary people and office workers use leading to the fact that the software development market began to focus on mass demand, where it was necessary to solve not engineering, but user and business tasks.

For example, one of these tasks is to create a text editor. Before MS Word began to dominate, a lot of trial and error was done, a lot of processors and text editors appeared, many of which only few people remember now: WordStar, WordPerfect, Lexicon, XyWriter, Lotus notes, Makurita and others. This kind of product development has to be conducted with a constant attention to user's reaction, quickly responding to his needs with new versions.

Approaches to development based on the cascade model, with clear sequential stages, and comprehensive documentation have received the conditional name of "heavy methods" **(heavy methodologies)**[3]**.** At the same time, various specialists from the IT world attempted to work differently, inventing new tools, techniques, or even developing whole methodologies in order to increase software adaptability and development speed as well as deliver valuable product to the user faster.

These new approaches used a similar **set of principles and tools:**

- iterative-incremental development;
- regular customer or user feedback;
- teamwork;
- maximum transparency.

Several so-called "lightweight" methods (lightweight methodologies)[4] emerged in the 90s of the 20th century: Crystal Clear, Extreme programming (XP), Rapid application development (RAD) and Dynamic System Development method (DSDM), ICONIX, Scrum[5], Adaptive Software Development (ASD), Function-driven Development (FDD).

Jim Highsmith, author of the book "Adaptive Software Development" recalled that time: “I think that at that moment we were all looking for legitimacy. Each of us could worked alone and did similar things in own way, but it could not have success and recognition in the community [of software developers].”

Classical **"heavy-methods"** were often officially fixed as standards for large-scale software development projects (for example, in the US Department of Defense). **"Lightweight methods"** have long been perceived as exotic, or specific ways of working in a specific company

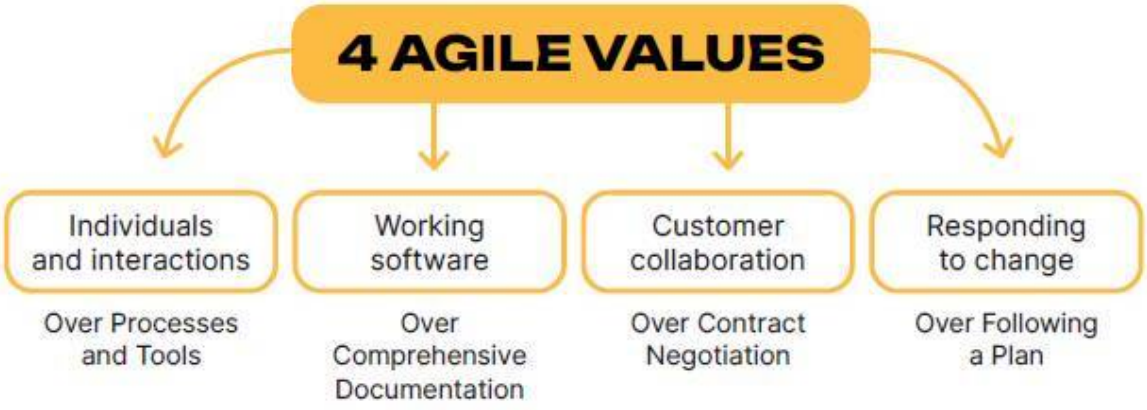
on a specific project. They did not receive wide recognition, and were promoted only by their founders' efforts.

Many people came up with the idea that it was necessary to organize a general meeting at which various "lightweight" methodologies founders and supporters would form a common document proclaiming a new software development paradigm. Then they could act as a united front, as an organized force, as opposed to "heavy-methods" dominance.

In early 2001, 17 people gathered not far from Wasatch Mountains in Snowbird, Utah, to discuss the future of software development. The participants of this group were united by concern about the current state of affairs in the industry, when "heavy-methodology" driven projects increasingly failed and flexible approaches were in great demand of legitimization and recognition. At the same time, they were not afraid that their thoughts about optimal solution differed.

The long weekend meeting resulted in **Agile Manifesto of Software Development** and became an answer to all these questions. This concise and expressive document consisted of only 68 words and changed software development forever. For almost two decades since its creation, these words (and the 12 principles that followed) have been adopted (to one degree or another) by a huge number of people, teams and companies.

Agile emerged as a mindset, a thought process that involves understanding, collaborating, learning, and staying flexible to achieve high-performing results, as a counterbalance to outdated approaches and excessive bureaucracy in IT field. Silicon Valley Residents realized that it is impossible to create innovative products in a conservative environment.



Here are Agile values with more detailed explanation.

There are four of them:

• if you want to build an agile process, you need to interact and communicate with each other. You can (and definitely will) use some tools, for example, trackers – JIRA, Redmine, etc. But the whole process should be based on various meetings and interaction, and not on tracker's settings or TFS (Microsoft stack);

• the working product that we make is much more important than the documentation for it. An example was given above with two companies. Documentation, the user cannot apply because the product is not ready, does not bring value to this user. If we learn to work minimizing software development steps, or by making them smaller, then we will have a more flexible process;

• cooperation and interaction with the customer is more important than strict contract following. It is usually named Fixed Price when you sign an agreement and fulfill agreed amount of work. The time, amount of work and deadlines are fixed. This approach is not very good if you want to work for the long term and be flexible. To be agile, it is more correct to build partnerships with the customer. The most important thing is that the search for a partnership and a win-win situation begins here, when both the customer and his contractor win;

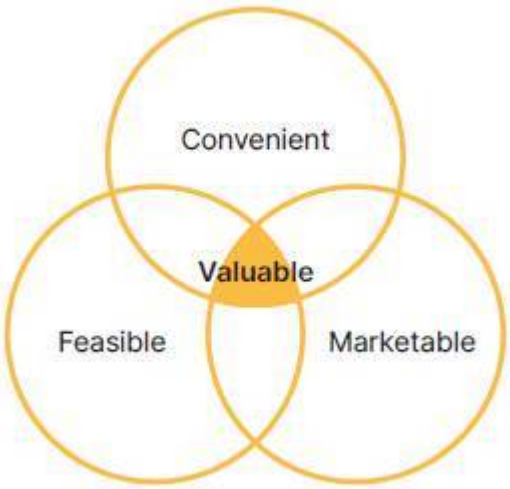
• readiness for changes with following the original plan. Agile development approach requires plan, estimates and forecasts. If you have an initial annual project plan and provide some working product version in some months or requirements change while developing you can change the whole plan taking changes and feedback into consideration.

12 Agile development principles, also the result of the Snowbird meeting, expand these several value-defining proposals.

It's all. Since then, the website with the Agile Manifesto has hardly changed (or maybe it hasn't changed at all), which can't be said about the world around Agile.

Agile methodologies have enjoyed overwhelming popularity: first mentioned in the PMBOK (the US Project Management Body of Knowledge standard) 5th edition (2013), then they were fully adopted in the PMBOK version 7, released in 2021 which took all the Agile principles and became its direct ambassador.

To conclude with Agile introduction, it is necessary to state **the main goal of this approach** – delivering value to the consumer. According to Agile methodology, it is achieved using **three characteristics** for a software product:



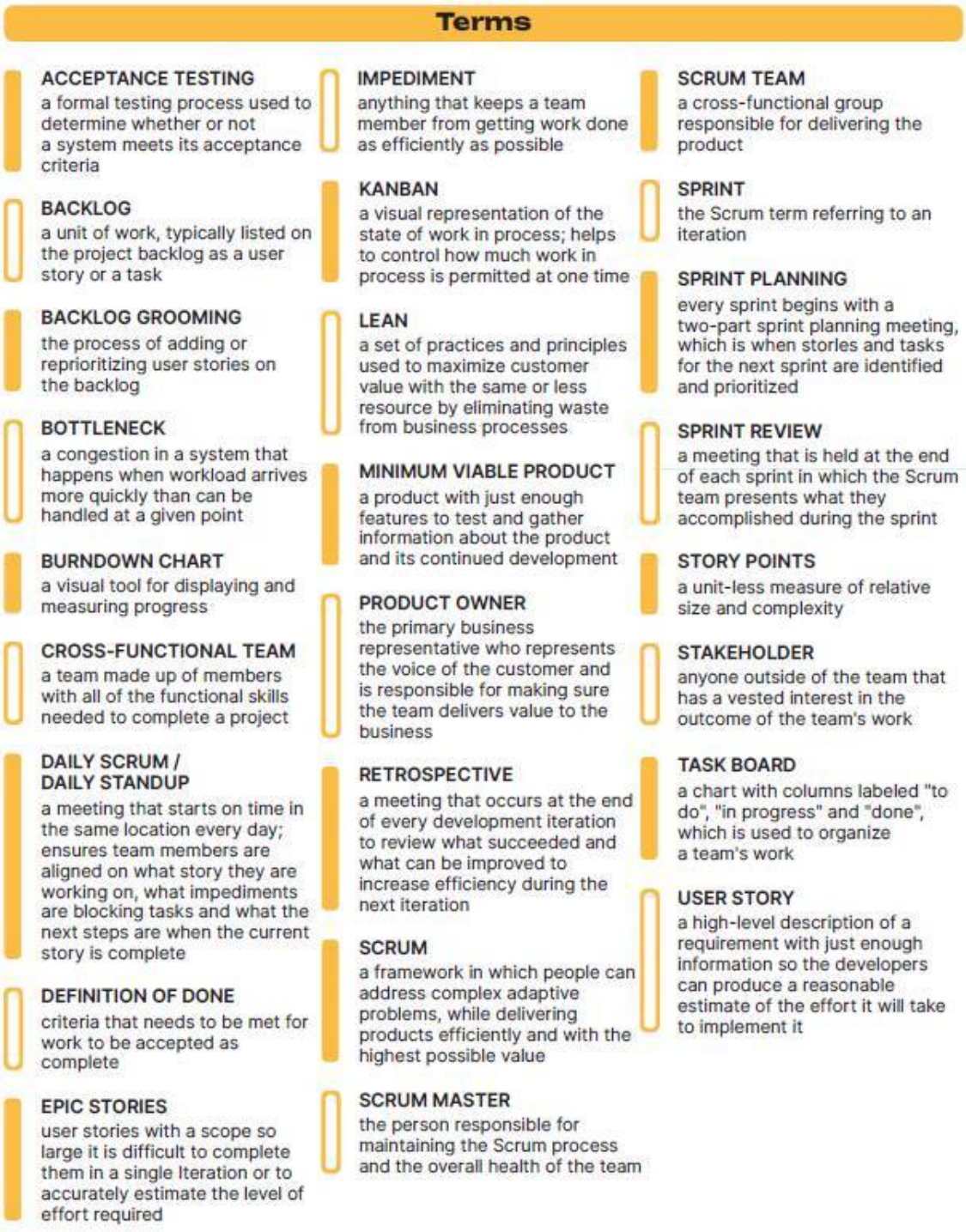
Thus, **the Agile method is applied to:**

• **accelerate the product launch to the market.** If you want to develop software faster, you need to apply Agile. For example, two similar business companies. The first one creates the technical task for software development, designs the structure and interface consistently, this is a waterfall model, implementing it can take several months. Another

team can already release a website and software applying Agile, start earning money and hijack the market;

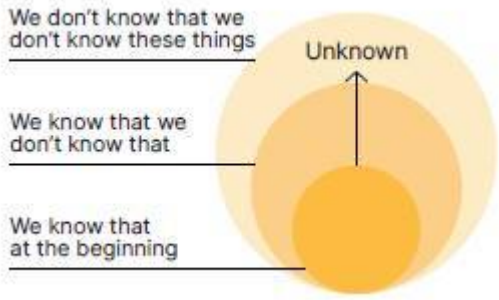
- **manage priority changes.** An unpleasant challenge for almost all companies. Your requirements will definitely change if you are doing the project that lasts at least a few months.What about commercial development, the problem is that programmers, analysts and designers never know what customer who pays us and users need. The usual approach is: until the user tries the site or application, you do not know whether it is needed or not;
- **improve cooperation between IT and business.** This is a headache, especially for large companies. Business requirements change from time to time, everyone speaks their own language. As a result, the parties do not understand each other.

Before diving into Agile, **it is important to learn terminology** in order to lay the foundation for our understanding of this approach. We will analyze all the terms in more detail during the dive, so do not be afraid that at the very beginning they will seem abstract and difficult to understand:



The very idea of Agile is to learn how to work with the unknown and fill blank pages of our work due to an iterative approach (working on short sprints[6] of 1-2 weeks).

This helps not only to keep up with the industry, but also to meet customer expectations and remain competitive.



Implementing Agile while building a software development process we start revealing practices:

• first of all **Stand-Ups (Daily Stand-Ups)**[7]. This is an easy practice: your team should meet regularly to synchronize the development process. Communication is key to developing effective teamwork, and poor communication can be one of the biggest reasons why a project fails in Agile teams. A daily stand up is a part of the agile methodology because it helps with improving communication among team members. Integrating effective communication with face-to-face contact as a part of the team's culture helps in creating a more Agile workplace;

• the second is **Sprint Planning**[8]. Your team should plan the work for the nearest iteration. It is an iterative approach, allowing teams to accelerate the delivery of value to customers and avoid unnecessary headaches. Instead of releasing the entire product as a whole, the agile team performs the work within small but convenient increments. Requirements, plans and results are constantly being checked for relevance, so that teams can quickly respond to changes;

• **unit-testing**[9]. This is a software testing method by which individual units of source code (sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures) are tested to determine whether they are fit for use. Unit

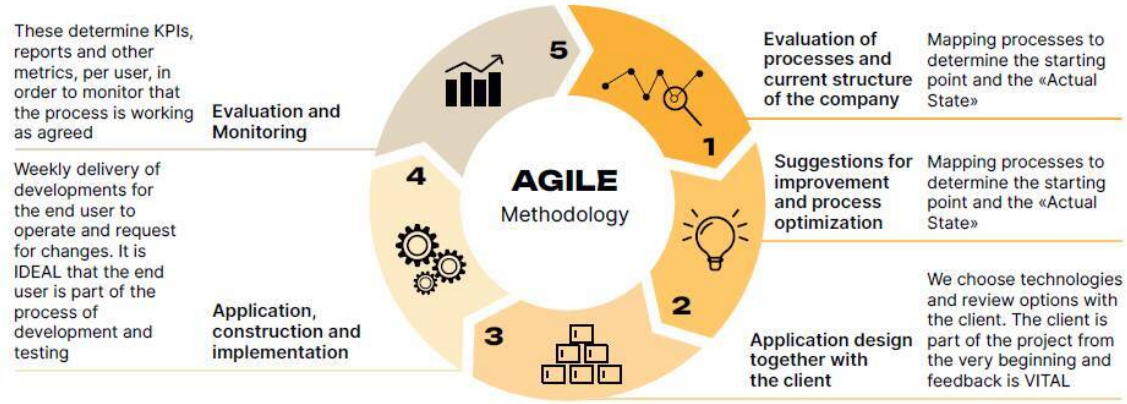
tests are typically automated tests written and run by software developers to ensure that a Chapter of an application or a program (known as the "unit") meets its design and behaves as intended. This is one of the simplest engineering practices that you can start using quickly. About 60-70% of bugs can be caught by unit testing;

- **Release Planning**[10]. We plan it in large parts: what and when we will release. Agile release planning is a product management method where you plan incremental releases of a product. It differs from raditional software planning where you focus on major releases. Using a release plan helps you plan which product increments (versions) get released to the market and when. And it's an integral part of the Agile SDLC (Software Development Life Cycle) because it can also give higher-ups peace of mind that there's a structure and plan beyond just the next sprint, and helps the individual Agile teams stay on track. What about Scrum, we can measure it with sprints (sprint planning).

It's no secret that change is hard and can be difficult to be taken, especially with complex projects. Making even slight changes to existing systems can often feel laborious and not worth the effort. Studies show (Identifying the Reasons for Software Project Failure and Some of their Proposed Remedial through BRIDGE Process Models. INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING. January, 2015, 3(1):118-126) that 60-80% of project failures can be caused by poor requirements gathering, analysis, and change management. The agile mindset is the best approach to variable and challenging environment as it offers an opportunity to embrace change, rather than continuously avoid it. It isn't something that teams achieve, but rather something that is continuously cultivated. Developing teams should strive to optimize, solve problems, reflect, and continuously improve in the process.

Both Agile principles and values contain a very correct message, but they look quite abstract from the first sight. Agile methodology

practices add to understanding of this approach. In order to investigate how Agile works, let's turn to the diagram and take a detailed look at the meaning of each link in Agile software development process:



LET'S SUMMARIZE WHAT WE'VE LEARNED IN THIS CHAPTER:

in brief, flexible methodology emergence was as natural as IT industry development. Deviation from out of time methods and software development flexibility request were influenced by huge expansion of the IT market and the subsequent demand for faster value to the consumer delivery.

Besides, now we know:

• Agile methodology started from 4 values which were followed by 12 principles;

• An iterative method of software development is in the base of agile approach;

• It gives developers an opportunity to deliver value to the consumer faster.

CHAPTER 2. WHAT ARE THE OTHER METHODOLOGIES? LET'S COMPARE WATERFALL AND AGILE

Let's see what approaches also exist in the world of software development besides the agile approach. First of all, this is the previously mentioned cascade methodology, also called "waterfall". We have already learned that in 1970 Dr. Winston Royce **defined Waterfall methodology and suggested its basic principles:**

• Program design comes first. Program designers work on the design of the system, not developers. Designers allocate e data processing models: database scheme, resources, some kind of non-functional requirements, interface, etc. Then an overview document must be prepared. The document should be understandable, informative, and current. Each team member must have an understanding of the system, and at least one person must have a deep understanding.

• Document the design. Some (in fact most) developers may wonder how much documentation is needed. Royce gives a simple answer: “Quite a lot”. Moreover, he says: “If the documentation is in serious default, my first recommendation is simple. Replace the project management. Stop all activities not related to documentation. Bring the documentation up to acceptable standards.”

Why documentation is important? Royce gives some points:

• documentation is a way of effective communication;

• during the early phase, the documentation is the specification and is the design;

• during the testing phase, the documentation can be considered as acceptance criteria;

• the documentation help newcomers to understand the system better.

• Do it twice. If your system is original, you want to verify your ideas or concept. Royce recommends creating some kind of **MVP (Minimum Viable Product)**[11] of your product to check the ideas. Then, the second version of your product will be finally delivered. The first version may not be provided to final customers/users, but **stakeholders**[12] want to see a prototype of the product. It may lead to requirement changes, and it is an effective way of the development process. The final version of the product will be closer to market needs.

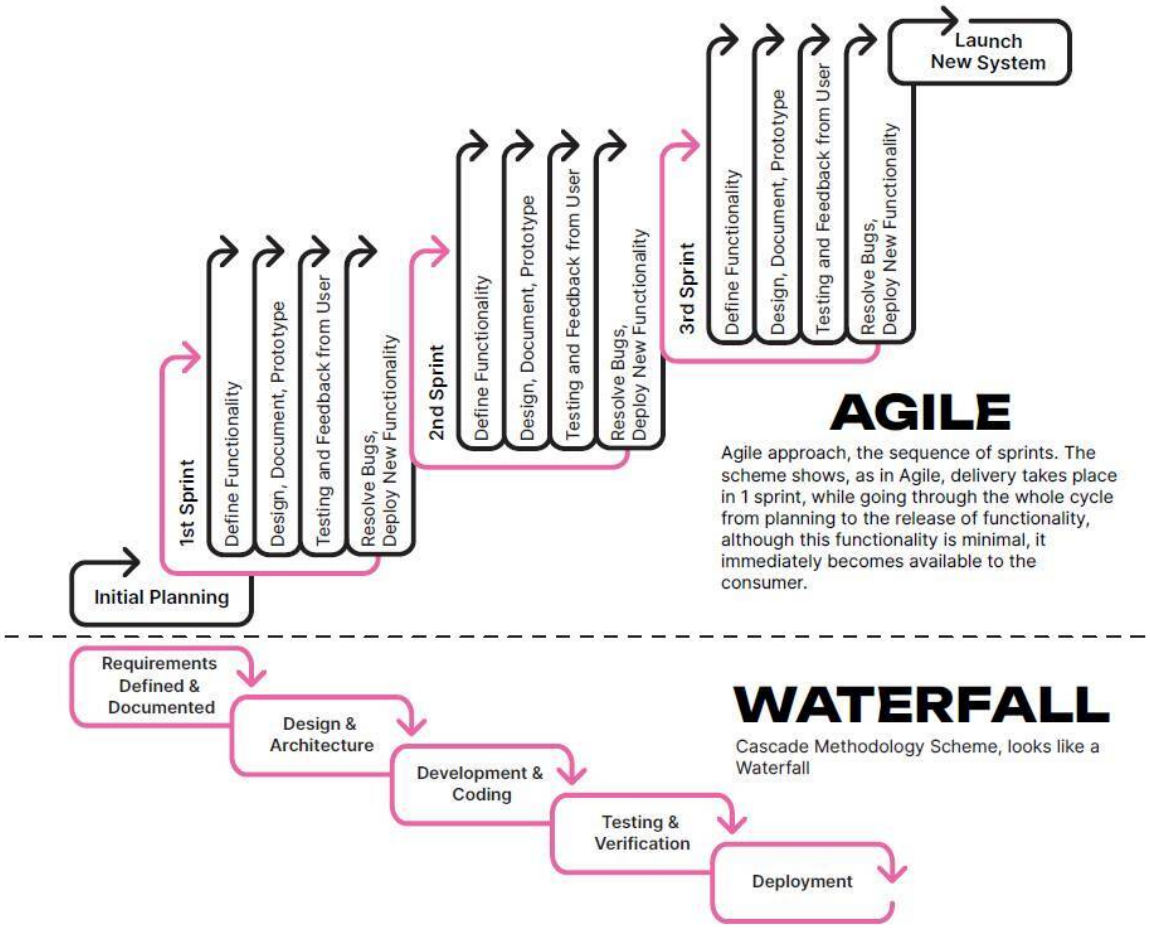
• Plan, control, and monitor testing. Testing is one of the most important phases of the **SDLC (Software Development Life Cycle)**[13]. You check your ideas and concepts, verify implemented system comparing it with system design. Royce recommends inviting the other persons to verify your system. He says that it would be more effective to give a big part of the testing to people who did not contribute to the system during the previous phases. The documentation will help them to test the system better.

• Involve the customer. Your project customer and stakeholders should be involved in all phases of the development. They will help to design and implement a system that will meet their needs closer. Moreover, the stakeholder may change their opinion about the system and give new requirements. As a Project Manager you should meet the changes with pleasure because after all your customer will be satisfied.

Waterfall is based on a logical sequence of steps that must be taken throughout the software development lifecycle. Each stage is coordinated by competent employees, documented and passed on.

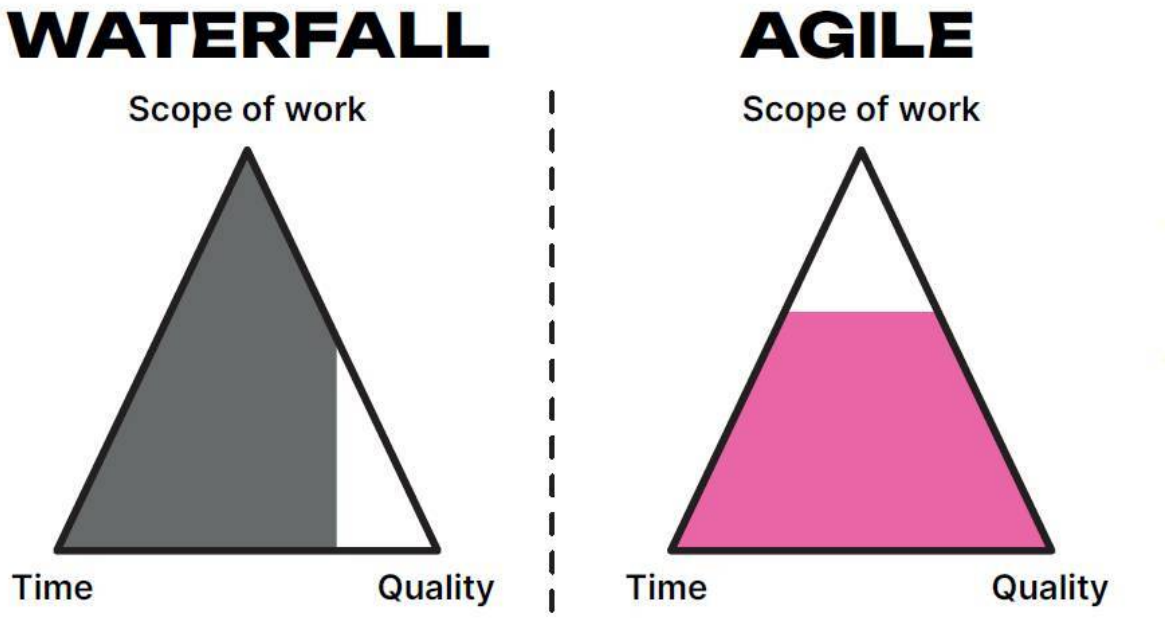
Although the popularity of the Waterfall model has waned in recent years, but the nature of the sequential process used in the waterfall method is intuitively close to developers.

The model proposed by Dr. Royce is extremely simple and clear. It consists of several blocks, each of them covers its own area of responsibility.



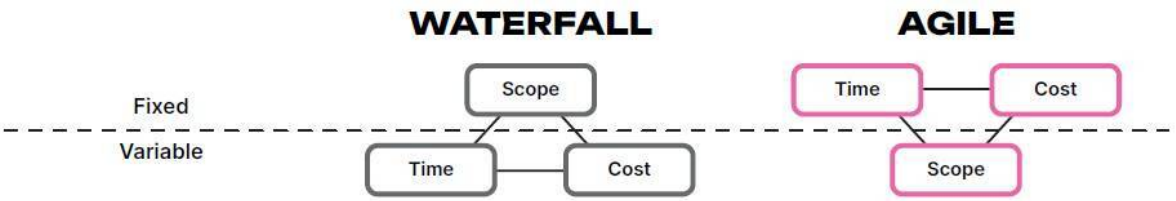
Let's look closer at **the difference between Agile and Waterfall methodologies.**

In management, there is a concept of a project management **triangle**[14]:



It includes the amount of work, time and quality. It is important to note that the balance in any methodology is created due to a system of assumptions, so in cascade approach quality can be reduced, and in Agile the amount of work can easily grow.

The project management triangle clearly displays the so-called **triple limitation** problem associated with the need to balance the scope of the project, its cost and time for implementation without compromising the quality of the final product. Taking into consideration the concept of project management triangle, **the difference between approaches** will look like this:



Focus	On processes	On people
Management	By plans	By changes
Development style	Waterfall / Long iterations	Incremental, iterative, short iterations
Requirements	Planned at the beginning	Updated before every iteration
Customer involved	At the beginning and the end	All the time
Feedback	Minimal	Frequent
Social aspects	Plan and control	Trust, responsibility, motivation
Team organization	Hierarchical	Self-organized teams
Leadership	Directing	Coaching
Change management	Minimization of changes	Accept and adapt to changes
Documentation	Exhaustive	Just enough
Communication	Mainly written	Mainly personal
Product Delivery	As a whole at the end	Frequent, but partial

In recent years, the waterfall model has been losing its leading position to more flexible methodologies. This is due to the general dynamics in IT, when teams of 5-9 people are responsible for software development, and the deadline can be easily shifted due to the increase in functionality.

Nevertheless, **the cascade model is still relevant for large projects and organizations:**

- It is resistant to personnel changes. Developers can come and go throughout the life cycle of the project, but thanks to detailed documentation, this practically does not affect the timing of the project;
- Waterfall model forces developers involved in the project to be disciplined and to stay within the plan. If necessary, a management body

responsible for decision-making is added to the general model, while performers are required to work within the system **framework**[15];

• Flexibility in the early stages. Changes in early phases can be made immediately and with minimal effort, since they are not backed up by code. Thus, the customer and the contractor have a significant time margin for a radical change in the concept of software work;

• Focus on deadlines and finances. Due to the fact that each stage completely outlines the contour of the future software, all developers understand their role, the boundaries of work and deadlines. This allows you to cooperate with the customer knowing the real cost of development and makes, therefore, the project model attractive.

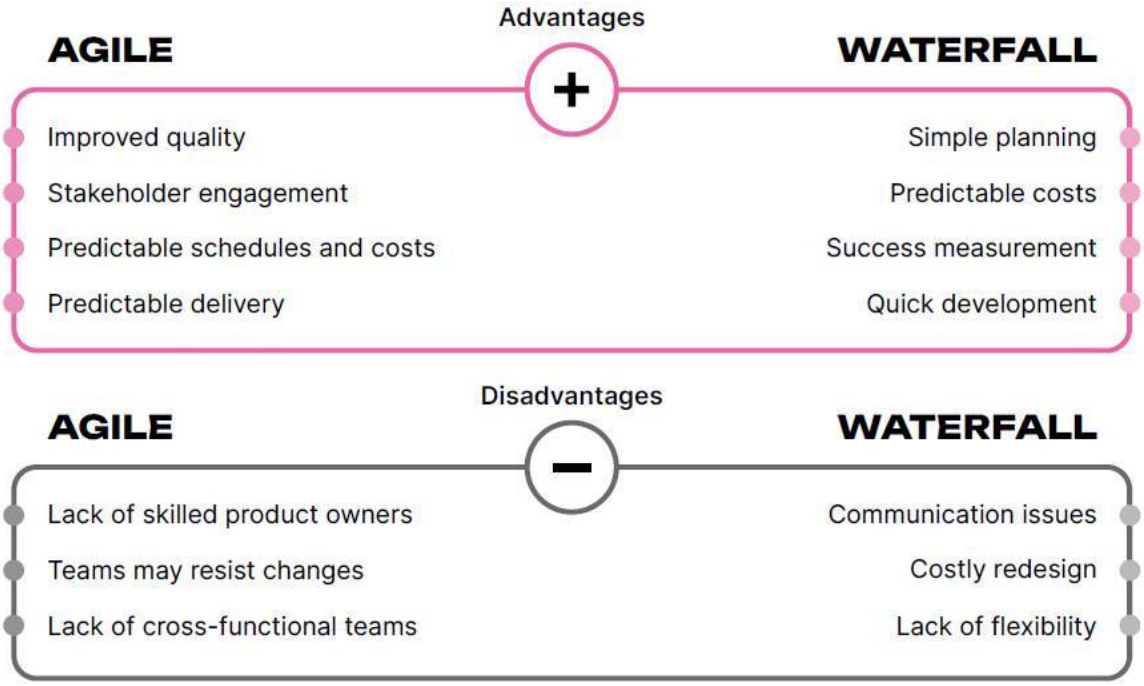
In the 1970s, Royce's ideas were relevant. But after almost 50 years, **the cascade development method is less and less suitable for IT:**

• non-adaptive software structure. At the first stages, the waterfall model can be flexible, but if problems in the overall structure are identified during the testing phase, this entails deplorable consequences in the form of disrupted deadlines and even customer failures;

• ignores the end user. The lower the process progresses in the waterfall, the less the role of the customer in it, not to mention the users he represents. Making any changes to the functionality of the software starts the entire chain of stages anew, so the products obtained by the cascade model are far from targeting the mass user;

• later testing. It is here that mistakes made at each of the stages are most often identified. More flexible methodologies use testing as a fundamental operation that occurs continuously. Waterfall, on the other hand, admits low qualifications of employees at each stage and poor quality of execution, because with late testing, problems cannot be solved fundamentally, only with the help of "patches".

Let's fix both approaches pros and cons:



Everything seems to be transparent, but in practice the question often arises which methodology to apply in each specific case. This scheme helps us mostly to make the right choice:

It turns out that **the cascade methodology is an excellent solution in terms of deadlines and reporting, but very weak in terms of quality.**

Despite the fact that these 3 points are increasingly rare in real practice, the cascade model will be popular and in demand for a long time because of its clear organization. This means that any professional developer should understand its basic principles and be ready to exist within the framework of this approach.

LET'S SUMMARIZE WHAT WE'VE LEARNED IN THIS CHAPTER:

Agile and Waterfall are the most common but opposite to each other software development approaches. While commending the waterfall methodology, it is necessary to move forward and look ahead, addressing contemporary challenges here and now. Agile is just what you need for this. We have also learned, that:

- software development methodologies should be evaluated using project management triangle;
- Agile methodology gained great popularity and changed software development forever by now;
- Waterfall or Cascade methodology is still in demand, so developers should be ready to exist within this approach.

CHAPTER 3. IMPLEMENTING AGILE APPROACH IDEAS INTO PRACTICE. LET'S BREAK DOWN THE INSTRUMENTS AND TECHNIQUES

Let’s look at frameworks that are used to implement Agile Approach ideas into real practice, but first let’s explain what a framework is.

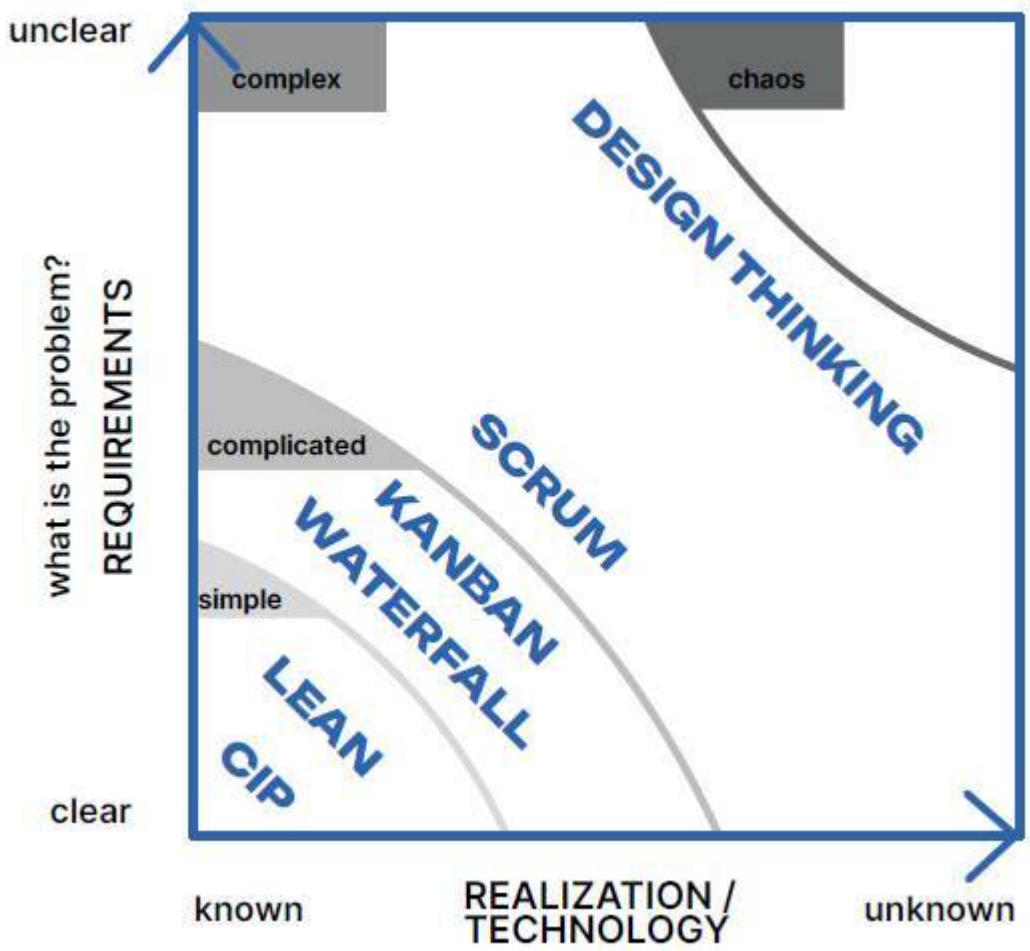
Framework is a ready-made set of tools that helps the developer to create a product quickly: a website, an application, an online store, a CMS system.

Imagine that you decided to assemble a toy car: it should drive and look almost like a real one. Obviously, there are several ways to solve this problem:

• the car can be assembled independently, from scratch: create all the parts manually or with machining equipment, do electronics and electrics, paint everything. So you can control the whole process, but it will take a lot of time;

• a similar car can be assembled from ready-made kits: body parts, engine, electronics will become modules that are important to choose and connect correctly. If desired, you can improve any module, but generally this is not necessary, everything will work right out of the box.

The framework is the same ready-made set, only for the developer. It saves the time and effort of a specialist who would have spent on creating basic things and correcting simple mistakes.



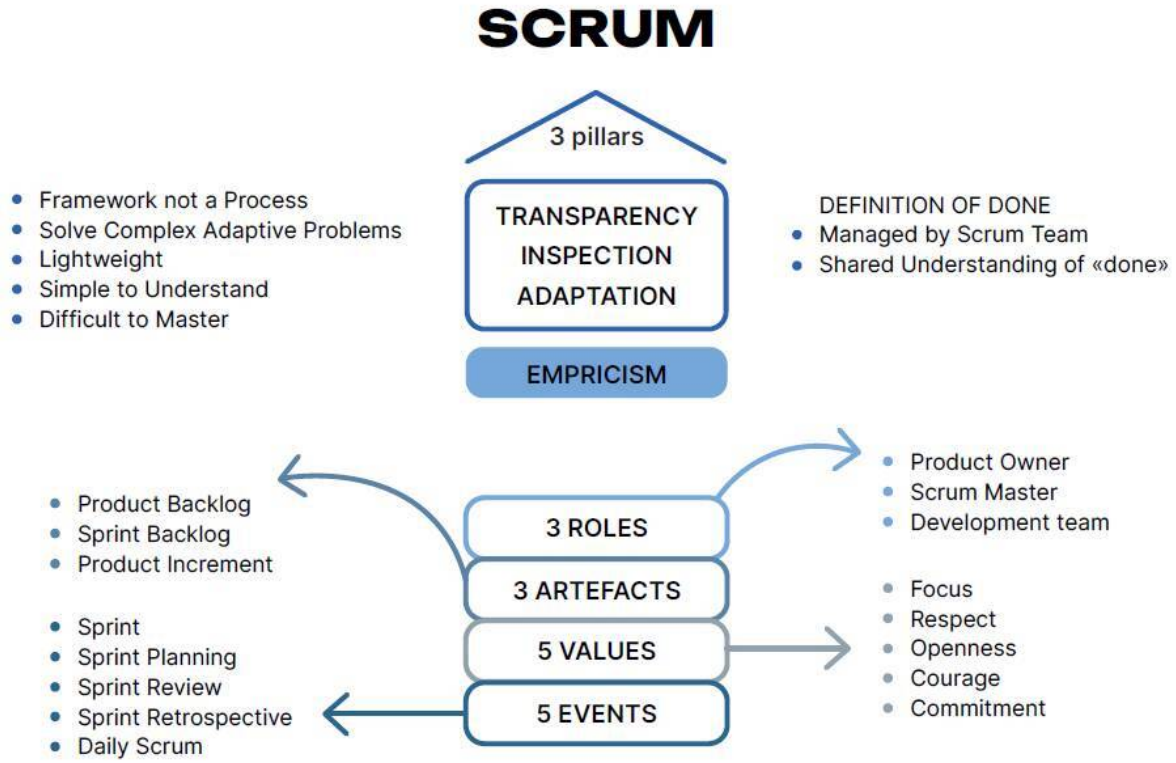
The most common Agile framework is Scrum.

PROCESS

Scrum projects are organized in short iterations, during which the team works to add incremental value to the end product.

Scrum is a set of rules allowing the team to establish a flexible workflow. Development is carried out in iterations, the goals of each iteration and the tasks of each team member are clearly outlined. Due to Scrum, companies can apply the principles and values of Agile and work on projects of any complexity. Agile and Scrum concepts are regularly confused, considering that Agile and Scrum are the same.

The difference lies in the scale of two approaches: Agile is a special way of thinking, a mindset. Scrum is an instruction for use. A clear plan describing each step of Agile implementing in product development. This is a methodology with specific stages, in which roles and events are clearly defined.



Scrum is based on empiricism and lean thinking.

EMPIRICISM asserts that the source of knowledge is experience, and decision-making is based on observations. Lean thinking reduces losses and focuses the main.

Scrum uses an iterative-incremental approach to optimize predictability and risk

management. It involves groups of people who collectively possess all the skills and experience to do the job, to share knowledge and acquire the necessary skills as needed.

Scrum combines four formal events for inspection and adaptation into a container event – Sprint. These events work well because they implement the empirical pillars of Scrum: transparency, inspection and adaptation.

TRANSPARENCY: the emerging process and work should be visible both to those who do the work and to those who get results. Important decisions in Scrum are based on an assessment of the state of three formal **artifacts**[16]. Artifacts with low transparency can lead to solutions that reduce value and increase risk. Transparency makes inspection possible. Inspection without transparency is misleading and is a waste.

INSPECTION: to identify potentially undesirable deviations and problems, it is necessary to regularly and thoroughly inspect Scrum artifacts and progress towards achieving agreed goals. To help with the inspection, Scrum provides a **cadence**[17] in the form of five events.

Inspection makes adaptation possible. Inspection without adaptation is considered meaningless. Scrum events are designed to provoke change.

ADAPTATION: if any aspects of the process go beyond acceptable limits, or if the final product is unacceptable, the process used or materials produced must be adjusted. The adjustment should be made as soon as possible to minimize further deviation. Adaptation becomes more difficult when people involved do not have authority or are not self-governing.

A **Scrum Team**[18] is expected to adapt the moment it learns something new during an inspection.

SCRUM VALUES

The successful use of Scrum depends on how much people share its **five values: commitment, focus, openness, respect and courage.**

A Scrum Team is committed to their goals and supporting each other. Their most important focus in working at Sprint is the maximum possible progress in achieving goals. The Scrum Team and stakeholders are open to discussing work and challenges. Scrum Team members respect each other as professionals and independent people, and in the same way they are respected by the people they work with. Scrum Team members have the courage to do the right thing and work on solving complex problems. These values set the direction for the work, actions and behavior of the Scrum Team. The decisions taken, the steps taken, and the way Scrum is used should reinforce these values, not weaken or undermine them. Scrum Team members comprehend and discover these values while working with Scrum events and artifacts. When these values are brought to life by Scrum Team members and the people they work with, the empirical pillars of Scrum – transparency, inspection and adaptation – come to life, building trust.

SCRUM ROLES[19]: TEAM (DEVELOPERS)

The main unit of Scrum is a small team of people, a Scrum Team. A Scrum Team consists of one **Scrum Master**[20], one **Product Owner**[21] and Developers. There are no subcommands or hierarchy inside the Scrum Team. This is a close association of professionals focused on one goal at any time given – Product Goal.

Scrum Teams are **cross-functional**[22], meaning their members have all the skills needed to create value in each Sprint. They are also self-governing, that is, they decide who does what, when and how. A Scrum Team is small enough to stay agile, and large enough to do significant work during a Sprint – usually consists of no more than 10 people. Overall, we found that small teams communicate better and are more productive. If Scrum Teams become too large, participants should consider reorganizing into several cohesive Scrum Teams, each focused

the same product. Therefore, they must have the same Product Goal, the same **Product Backlog**[23], the same Product Owner.

A Scrum Team performs all product activities: cooperation with stakeholders, verification, maintenance, operation, experiments, research, development and all that may be required. They are structured and empowered by the organization to manage their own work. Working in Sprints at a steady pace improves the focus and consistency of the Scrum Team.

The whole Scrum Team is responsible for valuable and useful Increment creation in every Sprint. **Scrum defines three responsibility zones in the Scrum Team:** Developers, Product Owner and Scrum Master.

Developers are the people in the Scrum Team who are committed to creating any aspect of a ready-to-use Increment in every Sprint. The specific skills needed by Developers depend on the subject area of the work being done and can be very different. However, Developers are always responsible for:

- creating a plan for Sprint - **Sprint Backlog**[24];
- striving for quality through compliance with the Definition of readiness (Definition of Done)[25];
- daily adaptation of your plan to achieve the Sprint Goal;
- mutual accountability to each other as professionals.

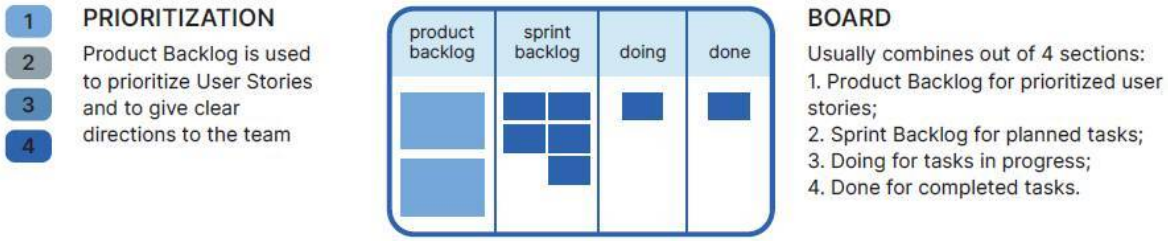
SCRUM ROLES: PRODUCT OWNER

A Product Owner is responsible for maximizing the value of the product resulting from the work of the Scrum Team. The ways to achieve maximum value can be very different and depend on organizations, Scrum Teams and specific people. The Product Owner is also responsible for the effective management of the Product Backlog, including:

- Product Goal developing and accurate communicating;
- Product Backlog elements creating and explaining;

• Product Backlog elements organizing;

• Product Backlog transparency, accessibility and understanding providing.



A Product Owner can do this work himself or delegate it to other persons. However, Product Owner remains responsible for it. In order for Product Owners to succeed in this, the entire organization must respect their decisions. These decisions are reflected in the content and order of the Product Backlog elements, as well as in the Increment being inspected during the **Sprint Review**[26]. A Product Owner is one person, not a committee. The Product Owner can reflect the needs of many stakeholders in the Product Backlog. Those who want to change the Product Backlog can do this by trying to convince the Product Owner.

SCRUM ROLES: SCRUM MASTER

A Scrum Master is responsible for applying Scrum in accordance with Scrum Guidelines. They do this by helping everyone understand Scrum theory and practices, both within the Scrum Team and organization. The Scrum Master is responsible for Scrum Team effectiveness, helping the Scrum Team improve their working methods within the Scrum framework. Scrum Masters are true leaders who serve the Scrum Team and entire organization.

A Scrum Master serves the Scrum Team in several ways, including:

• advising team members on self-management and cross-functionality;
• helping the Scrum Team focus on creating Increments with high value that meet the Definition of readiness;
• helping to eliminate obstacles that hinder the progress of the Scrum Team;
• making sure that all Scrum events occur, are positive, productive and do not go beyond the time limits.

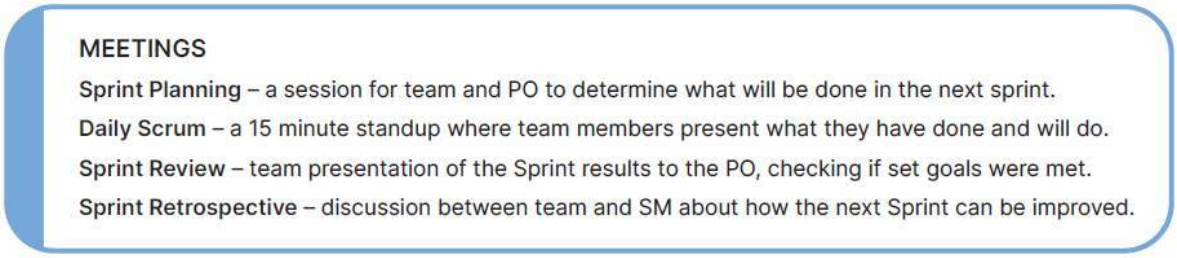
A Scrum Master serves the Product Owner in several ways, including:

• helping to find techniques for effective Product Goal detection and Product Backlog management;
• helping the Scrum Team realize the need for clear and concise Product Backlog elements;
• helping to apply empirical product planning in a complex environment;
• facilitating interaction with stakeholders upon request or if necessary.

A Scrum Master serves the organization in several ways, including:

• directing, training and advising the organization in Scrum application;
• planning the transition to Scrum and advising on Scrum application within the organization;
• helping employees and stakeholders understand and apply an empirical approach to complex work;
• removing barriers between stakeholders and Scrum Teams.

SCRUM EVENTS



Sprint is a container for all other events. Every event in Scrum is a formal opportunity for inspection and adaptation of Scrum artifacts. These events are specially designed to provide the necessary transparency. Failure to conduct any of the Scrum events in accordance with the description leads to a loss of opportunities for inspection and adaptation. Events are used in Scrum to create regularity and minimize the need for meetings not defined in Scrum. A good practice of reducing complexity is to hold all events at the same time, in the same place.

SPRINT

Sprints are the pulse of Scrum, where ideas turn into value. This event has a fixed duration of no more than one month to ensure consistency. A new Sprint starts immediately after the completion of the

previous one. All the work needed to achieve the Product Goal, including Sprint Planning, Daily Scrum[27], Sprint Review and Sprint Retrospective[28] events, is done within Sprints.

Конец ознакомительного фрагмента.

AGILE TRANSFORMATION

In IT Organizations

For managers of IT teams, developers, and executives in the information technology industry

The most important attitude that can be formed is the desire to change.

Alisa Bobovnikova

Примечания

1

Waterfall methodology, also called cascade or waterfall, is a classic model of the software development lifecycle.

Вернуться

2

Combination of both iterative and incremental software development methods when the development progress is achieved by short time periods (iterations), each of them delivering an increment to the product development process.

Вернуться

3

Approaches to software development based on the cascade model, with clear sequential stages, and comprehensive documentation.

Вернуться

4

Approaches to software development based on the iterative incremental method.

Вернуться

5

The most common and popular Agile framework.

Вернуться

6

A short time period during which a scrum team performs a given amount of work.

Вернуться

7

A short meeting from 5 to 20 minutes a day, at which team members tell share what is happening in their work tasks.

Вернуться

8

The meeting that starts each sprint, and is intended to determine the team's work plan throughout the sprint.

Вернуться

9

This is a software testing method by which individual units of source code are tested to determine whether they are fit for use.

Вернуться

10

This is a product management method in which you develop a strategy for its phased release.

Вернуться

11

The product or service test version with a minimal set of functions (sometimes even one) that delivers value to the end user.

Вернуться

12

A person or an organization that has rights, interests, requirements or interests regarding the system or its properties that meet their needs and expectations.

Вернуться

13

The information systems creating concept, including their planning, development, testing and deployment.

Вернуться

14

This is a model, illustrating the three constraints interdependence: time, money and scale, as well as how changing one factor requires changing others.

Вернуться

15

A ready-made set of tools that helps a developer to quickly create a product: a website, an application, an online store etc.

Вернуться

16

This is the information with which the Scrum team and stakeholders describe in detail the product being developed, as well as the actions to create it and the activities within the project.

Вернуться

17

An Agile cadence is a reliable series of events and activities that occur on a regular, predictable schedule.

Вернуться

18

This is a small group of people, within which there is no hierarchy, mini-teams or leader and all participants work on the same goal.

Вернуться

19

A Scrum team participants' roles.

Вернуться

20

A Scrum role which does not imply anything other than the correct Scrum process conduction. Thus, the Scrum master is the server-leader of the team.

A Scrum role, a person responsible for the product creation and management. (S)he monitors its development process, records the necessary

metrics and is responsible for the result.

Вернуться

21

A Scrum role, a person responsible for the product creation and management. (S)he monitors its development process, records the necessary metrics and is responsible for the result.

Вернуться

22

A team that has all the skills necessary to do the job and is independent on those who are not part of the team. Cross-functional Teams are more flexible, creative and productive than teams where people are specialized in one competence to do their job.

Вернуться

23

An ordered set of elements, a queue of tasks, a list of all the functions that you want to get from the product. This list contains brief descriptions of all the product features desired.

Вернуться

24

A list of specific tasks to implement the product backlog selected elements.

Вернуться

25

This is a list of process and increment conditions under which the backlog item can be considered done.

Вернуться

26

The meeting at the end of each sprint intended to present the product developed by the team during the sprint.

Вернуться

27

This is a Scrum meeting, one of 5 Scrum events, that lasts no longer than fifteen minutes and is held every working day in the same place at the same time.

Вернуться

28

This is a workshop which is held to discuss how to improve the workflow.

Вернуться