If you are a novice to programming or business working with software house, you may imagine the software development process as a start-to-finish task. The client gives the developers the instructions for the end product, and they simply create it. Once it is ready, the client receives it and pays for the service. It is the idea behind most product developments. However, when it comes to software, the whole process is more efficient. How can any process be more efficient than the above, you might ask? It is, therefore, worth elaborating a little bit more about what software development looks like. Today, I will present the best methodology to ever grace the Earth. It may be a subjective opinion, but you are free to form your own judgment based on the below facts. None of what follows is a lie!

## Crucial assumption to begin with

**This article by definition does not cover every detail of every topic that is mentioned inside of it. This is an intro to Agile as a concept and an intro to  certain Agile frameworks that may be useful in software development (Scrum, Kan-Ban & ScrumBan). But there are other articles on this blog tagged “Agile for busniess” that look into perspectives of this topic.**

**Last but not least – please look deeply at the last section of this document (“Useful knowledge sources – books/essays/websites/YT channels etc. “) where you can find multiple knowledge sources. Some of them (books) are in Sailing Byte’s library rest is accessible completely free on the internet!**

## Why is Sailing Byte trying to learn, implement (if needed) and share knowledge about Agile?

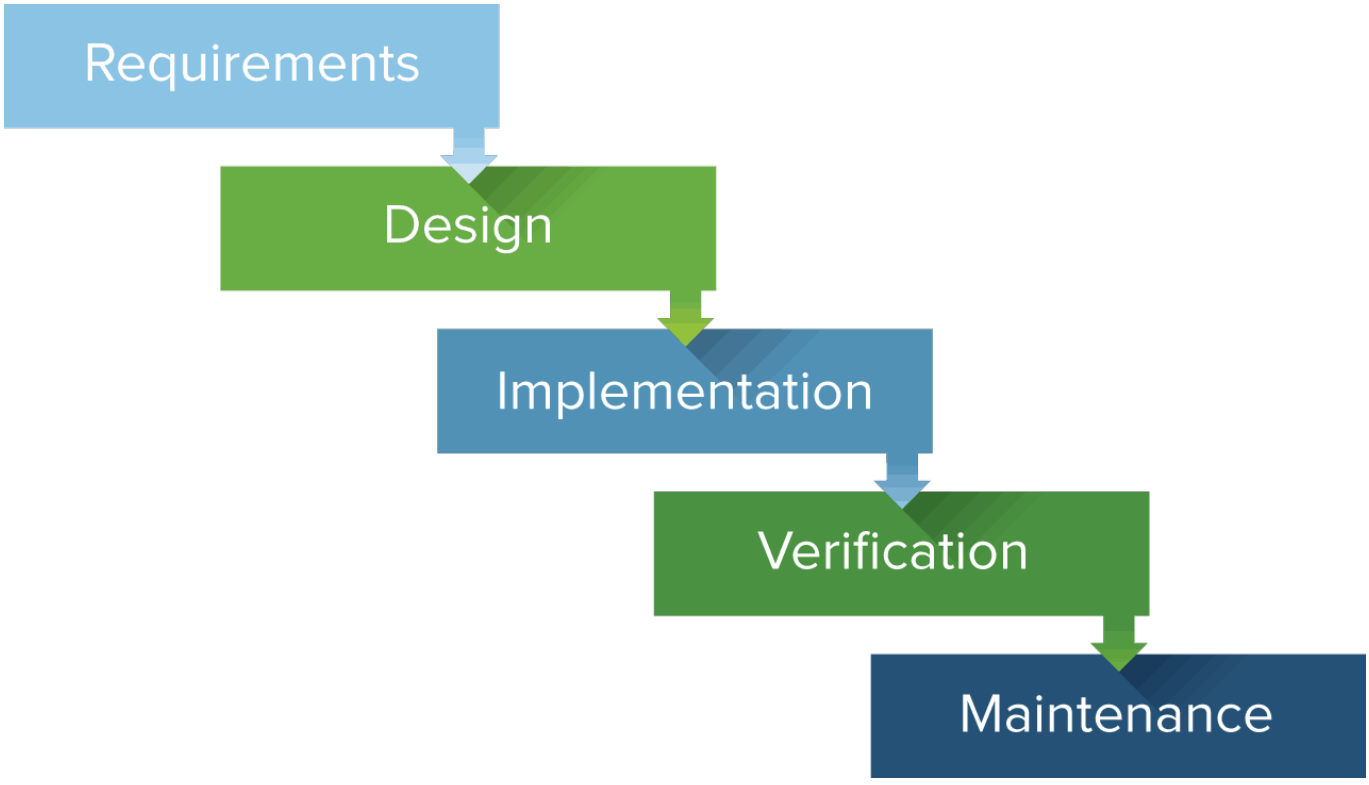
**Sailing Byte it’s under constant processes of Scrum/Scrumban improvement. We have recognized that there are potential organizational benefits in introducing agile frameworks to our ongoing software development projects.**

**Ultimately we want to deliver high quality code (therefore more value to end customers) with a more predictable pace through the whole portfolio of projects. To obtain that goal we may have to change the way we work and cooperate with certain stakeholders. We are still learning how deeply we can apply those changes in various projects.**

**Sharing knowledge about Agile methodologies to both existing & future Sailing Byte clients works only beneficial. It's a clear win-win situation when both sides understand from the very beginning how the whole process looks like and what are the benefits of a certain approach.**

## What does "Agile" mean in context of software development?

Term agile means to be *"able to move quickly and easily."* World of software development of 80' and 90' was far from being able to quickly and easily manage any changes. Every business need for new software had to follow a strict "Waterfall" path of development




There is nothing bad by definition with this (Waterfall) approach. It's perfectly reasonable to work all stepps (requirements, design, implementation, verification, maintenance) one by one. For example when you would be asked to build a bridge there would be nothing bad in first gathering requirements (who needs that bridge and for what), then designing it (making a strict technical plan) then implementing that plan exactly as planned (start construction work). Finally you will have to cover

verification (check if that bridge is safe and serve its purpose) and maintenance (make sure that bridge will not collapse and will be useful for end customers in the next 100 years).

Bridge has to be built with a **constant set of requirements** to **serve its needs** which can happen only if you will finally **build the whole** bridge. There is **no value** for the end customer in a bridge done 65% or even 99.9% To deliver value faster you **can't change** plans in the middle of construction work. It's rather obvious.

If you will apply this (waterfall) method to software development it will be clear that the end customer will have to wait for **value** until the very end of a whole process. You will have to know **all requirements** at the very beginning and you **will not be able to change them** unless you are ready to handle all the risks that come with a possible 2-3x times longer development period. (budget waste, waste of company production capacity, pressure from stakeholders etc.)

Now let's ask yourself if the world is changing now in 2021 faster than it Was changing in 80'-90' ? Do customers change their behaviours and choose different applications to solve their problems/needs more frequently ? Do product's have to change more frequently due to that "world acceleration" ?

**Answers to those questions are rather obvious but imagine that there was a brave group of software engineers who saw that the world is "accelerating" way faster than the others. Those American & English software engineers proposed in february 2001 "Manifesto for Agile Software Development" (later "Agile Manifesto") – a bald statement that changed the future of software development.**

## Manifesto for Agile Software Development – What Does it State?

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation

- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Moreover that group of engineers proposed a list of 12 **principles behind the Agile Manifesto.**

In 2001 a group of developers, who had been following a new approach for a while, came up with the Agile Manifesto. The set of principles documented their shared beliefs and experience of how a modern software development process should look. The Agile Manifesto was signed by Ken Schwaber, Kent Beck, Martin Fowler, Ron Jeffries, and Jeff Sutherland. It exists to this day on the official website of the Agile approach.

Agile believes in the importance of collaboration, self-organization, and the ability to adjust to constant change. It is opposed to an overcomplicated documentation process, rigid management, and strict rules. People work with the main purpose in mind – the fast creation of a high-quality product.

> *We follow these principles:*
>
> 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. Agile processes promote sustainable development.

8. 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.

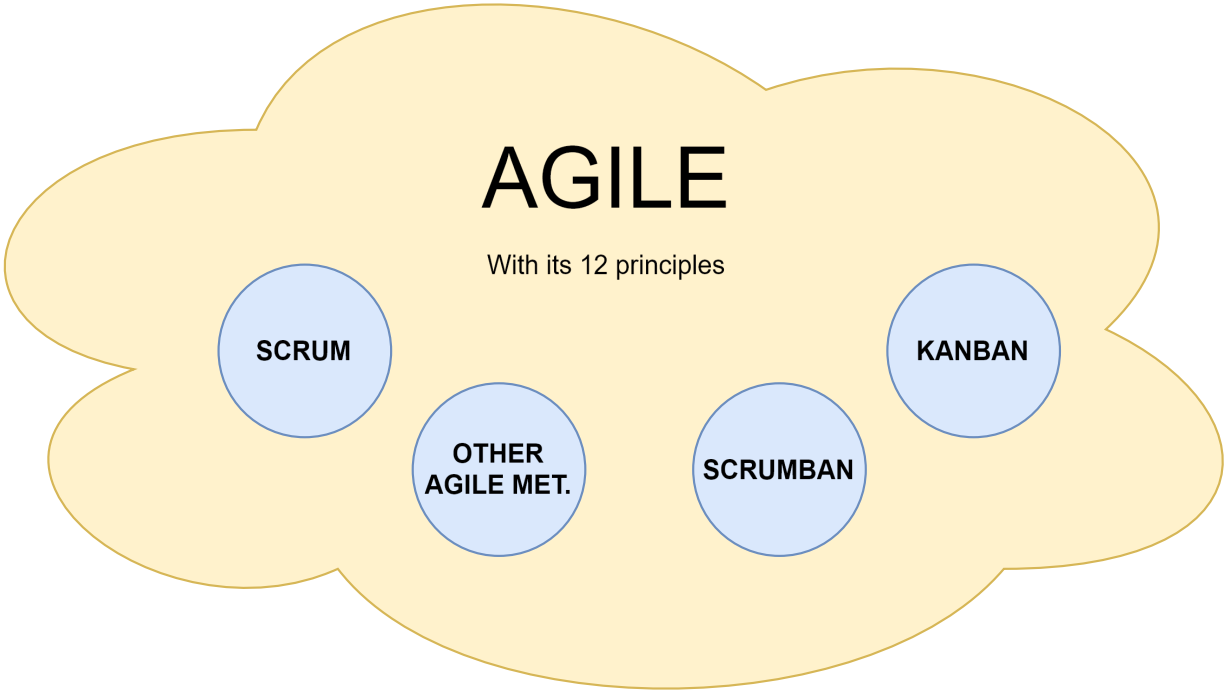
***more about Agile Manifesto:***
*Agilemanifesto.org*

Combination of such a manifesto and it's principles had a tremendous impact  on the world of software development. There was an urge inside of industry "biggest brains" to get rid of "waterfall" as the **only way** to produce software.

Approach that they proposed gives a main advantage in the form of faster **delivery of value** to the end customer combined with faster **feedback.** As a result we are now able to lower the risk that comes with delivering a project that at the end of the development process **will NOT cover end customer needs.** Those needs can change at any time and they will change only more frequently in the 21st century.

# What Are Agile Methodologies?

Let's start with the basics. Agile methodologies are approaches to the management of software development. There are several of them, but their main characteristic is the existence of general project assumptions, which get gradually adjusted during the stages of development. These adjustments come from the client, who reassesses their requirements based on the effects of the work.




# Scrum vs Agile – What Is the Difference?

While speaking of Agile, you may have come across the term Scrum. While Kanban and Scrumban are not very often mistaken with Agile, for some reason Scrum is sometimes used as synonym for Agile. But that is not the same. So what is the Scrum definition? It is one of the frameworks (the ways of conduct, tools) that can be used in Agile methodology. So, Agile is a movement, the general idea, and Scrum is one of the methods of fulfilling the Agile objectives. Many treat Agile and Scrum as different approaches. However, the Scrum vs Agile concept should not exist. Scrum derives from Agile, and both have a common aim, which is fast and high-quality product development.

## What is Scrum?

Knowing the Scrum definition, we can now focus on the Scrum meaning. Scrum divides the project into small parts. These are the phases during which the team develops certain functions. These stages are called Sprints and last roughly four weeks. The changes introduced during Sprints should be valuable improvements

visible to the software users. There are other interesting aspects of the approach, such as:

- the Product Owner’s role consisting of collecting the initial requirements needed to start the sprint and controlling the adherence
- Sprint planning – selecting the goal of every phase and the tasks with the highest priority to work on first
- conducting short daily meetings (Daily Scrums) where the team discusses the work performed the previous day and the work to be performed on the day of a meeting
- The Sprint review planned after each sprint sums up the work, concludes, and sets the ground for improvements.

## How was Scrum created?

Scrum meaning was introduced in 1986 by Hirotaka Takeuchi and Ikujiro Nonaka in a Harvard Business Review article. They emphasized two characteristics of Scrum (flexibility and speed) and compared the overlapping phases of product development to playing rugby. They stressed the importance of teamwork and striving to achieve better results. However, it was only after some time that the implementation of the Scrum meaning in software development happened. Ken Schwaber and Mike Beedle described the Scrum methodology in ‘Agile Software Development with Scrum’ in 2001. Schwaber also founded the Scrum Alliance (2002), Scrum.org (2009), and published The Scrum Guide, which defines the Scrum framework. The document is updated if necessary, according to the purpose of Scrum – strive to achieve better results.

## What is Agile?

Agile, as mentioned in the introduction, is a set of methods based on the adjustment to ever-evolving client requirements. It is used to quickly develop the highest quality product. A crucial part of the Agile methodology is the structure of the teams who work on the project. Usually, there is a lack of any organizational hierarchy. The team is cross-functional and self-managed. The members can decide how the set goals will be achieved. Communication between the members is vital to the project’s success.

### How did Agile start?

Agile was proposed as a change from the waterfall methodology. The development of Internet applications was the moment when the developers started to search for a more efficient methodology introducing iterative and collaborative processes. All that was due to the increased workload and the varied background of many small teams that worked on demanding projects. The best part of Agile is that it came naturally. In other words, no one sat down one day and said: '*The waterfall approach is no good for creating software anymore. From now on, the developers will work according to the following…'.* Far from that! Agile developed as a result of the developer's work. The very people who created software were the start of the new way of conduct, just because it allowed them to work better. That is pretty much the essence of Agile development – use the methods that help you achieve a better outcome in the fastest possible time. Does it mean there are no rules in Agile? Not at all. The rules are there. They are just not strictly imposed but come naturally instead. This set of rules is called the Agile Manifesto.

## Why are Agile methodologies better than the previous approach?

A simple answer, which is also in correspondence with the first rule of the Agile Manifesto, is that the Agile approach is simply faster and gives better results. Thanks to Agile, the developers prioritize what should be prioritized – the final product. They care less about how they develop it as long as the requirements are fulfilled. They know that the documentation and rules should help people at work. However, too much of the above will shift focus from the essence of that work. Also, splitting the work into manageable parts (Sprints) enables developers to complete it faster. They present the Sprint result to the client, and receive the new requirements. It is contrary to a long, scrumptious plan of all project stages that may even turn out to be fruitless as the project unfolds.

Finally, thanks to a more relaxed and goal-oriented approach, the developers are happier and better motivated. Who would not be happy if they could work and achieve success, without unnecessary fuss?

I cannot help but associate Agile with desire paths. In urban planning, people create desire paths because these are shortcuts or easier routes than the paved routes provided. I believe this is a perfect metaphor for why people use Agile in software

development instead of the waterfall approach.

**About frameworks/tools:**

Scrum Guide – version 2020
“Scrum with CanBan” scrum.org guide – version 2021
Labirynty Scruma – Jacek Wieczorek
Scrumban: Essays on Kanban Systems for Lean Software Development, Corey Ladas
Kan-Ban – Agile Project Management with Kanban, Eric Brechner
Succeeding with Agile: Software Development Using Scrum

**Inspirations :**
scrumgroup.org
Scrum.org
Zainspiruj mnie kuba – PL YT channel
AgileRebels.org
Agile.org
Atlassian – YT channel
Development that pays – YT channel
kanbanblog.com

**Books that are loosely connected with Agile/organization/management topics:**

Cargo Cult Programming – Richard P. Feynman
calteches.library.caltech.edu/51/2/CargoCult.pdf
The Machine That Changed The World – James P. Womack, Daniel T. Jones, Daniel Roos
It’s Not Luck – Eliyahu M. Goldratt (TOC)
THE GOAL – Eliyahu M. Goldratt (TOC)
Slack – Tom DeMarco
Project Phoenix – Gene Kim & Kevin Behr, George Spafford
Elegant Puzzle – An Elegant Puzzle, Systems Of Engineering Management – WIlliam Larsson
Five dysfunctions of a team – Patric Lencioni
Extreme Ownership – Jocko Willink  & Leif Babin

# Agile in everyday life – Sailing Byte as the experts in the approach

Today’s world is all about changes, fast adjustments, and improvements. It would be counterproductive to stay stuck in the waterfall approach. This is why, as soon as Agile Manifesto was published, it was well received by many software development companies all over the world. According to zippla.com, 86% of international software developers use Agile. Multiple well-known enterprises, such as Apple, Google, IBM, Cisco, Microsoft, and Spacex, utilize Scrum meaning and Agile approaches within their structures. Who else uses it? Sailing Byte, of course! If you want your product to be created within the shortest time possible and to the highest standards, Agile is the way to go. Book a call today to discuss the Agile and Scrum meaning in the development process of your software.