You may be confused why the same aspects are constantly mentioned as both positives and negatives during summary at the end of every article about Scrum, KanBan and Scrum-Ban. There is a purpose to it and that purpose is to always consider “it depends” as the best possible first answer to every question.

Scrum vs Agile vs Kanban or Scrumban? – A brief outlook on an ongoing Agile transformation

Many people get easily confused with the meaning of terms involved in Agile processes. That is why, before we move any further, let’s get on the same page with the terminology. Agile is the name of the methodology characterized by splitting work into stages within which we adjust that work to changing requirements. The idea behind Agile is turned into reality within different frameworks. Scrum, Kanban, and Scrumban are examples of those frameworks. So technically, such comparisons as Scrum vs Agile vs Kanban should not exist.

Why are there so many frameworks? The best answer to this question would probably be: because there are many ways in which people like to work. Agile believes in prioritizing effectiveness. That is why Agile transformation is an ongoing process. There is one mutual goal of all the frameworks. They lead to the development of the best possible product for the client. Let us illustrate this objective by going through the principles of Scrum, Kanban, and Scrumban, one by one.

Which Agile Framework is THE BEST (and why it is not possible to answer that question)?

Answer: “It depends on: “

1) What are you trying to achieve by applying a certain framework?

Maybe you just need a next round of “organizational changes” that will excuse the current not-so-good state of a certain product? I hope not. It will be way better to fully commit to a certain framework after careful analysis (please engage your developers into that analysys!) & live test on existing project.

2) What is the current structure of your product development?

Maybe you are already developing a product in a more-or-less agile way but you didn't even consider to name it that way. If there are only few things that need to change to adapt your current production/workflow to for example KanBan why to go around and force for example Scrum ?

The less changes you will force on the whole organization **at once** the better.

3) What is Your product and why do you even want to consider Agile as a valuable approach to that product development ?

As mentioned at the very beginning there are products (a bridge for example) that **need to be** developed in the waterfall approach. You simply will not benefit from applying Agile frameworks into bridge development. Maybe your product is not exactly a bridge but follows the same rule ? If yes then organizational effort to impose "Agile" will be a clear waste.

4) Do you have a vision for your product ?

It's almost impossible to imagine a product that will not benefit from having a clearly stated future plan for development. Even if those plans will change (and the truth is that they will change a lot) you are creating a common sense through the whole company that You as a product owner/CEO/product manager/project manager (choose which name you want) care about the product.

This is a very important signal to:

- developers – it gives them a purpose to focus on what they are best at – development! If they see that you have a clear product goal and plan to deliver it they will follow you!
- future employees – do you want it or not current team members are sharing opinion about your company/team **all the time**. Creating a product vision is partially giving you also a good PR on the job market.
- current & future business partners – if you are clear in expressing vision for your product you will be able to maintain those partnerships that are valuable to your organization.

5) Are you sure that the culture of your organization will align with Agile's principles

As stated a few times in this document changing culture of the whole organization is very difficult. Implementing any agile framework it's pointless if your organization

it’s not already at least a bit transparent and ready to inspect & adapt (thus ready to accept failure as a natural outcome of changing “how we work”). Even the best possible product to implement an agile framework will not benefit from it without proper values as foundations.

If you are running a small company (or one team or division of teams inside a bigger company) you have the biggest influence to impose cultural changes by **clearly expressing by yourself** all those values that you want to impose on your team members. If that will work out and you will find team members eager to follow those values (sometimes you will have to fire someone who is not willing to adapt) AND those values are in line with Agile’s values you should try to use an agile framework. Then it’s almost guaranteed that your team/small company will benefit from using a certain framework thus it will grow and create curiosity on the market/inside of a bigger company.

At this point you are:

- running a small company and you just have **a hard task** to maintain this “proper culture” through a period of painful company growth;
- running a team or division in a bigger company and you **have a very very hard task** of convincing upper management/CEO/board to apply your team’s culture upon the whole company.

What is Scrum?

As previously mentioned, Scrum is a framework used for the implementation of Agile methodology. The main characteristic of this framework is the breakdown of the project into time-bound increments, called Sprints. A Sprint lasts up to 4 weeks and finishes with the delivery of some pre-agreed part of the project. The main focus of the Scrum framework is to continuously improve the Sprints, the goal, and the overall product development process. There are other characteristics of Scrum, such as:

- regular planning occurs at the beginning of a Sprint
- estimations of time needed to perform tasks are done before the Sprint starts
- tasks to be completed within the Sprint should be small, if they are larger, they should undergo a further split
- any changes to the scope should wait until the next Sprint
- there are four main types of meetings: Sprint planning, daily Scrum, Sprint

review, and Sprint retrospective
- there are specific roles distinguished, such as Scrum Master, Product Owner, Development Team
- the ownership belongs to the Product Owner

What is Kanban?

This framework also divides the projects into small and manageable ‘chunks’. What is Kanban then, and why do we need it? How is it different from Scrum? Kanban does not focus on time as the most important factor in completing the stages. In Kanban, work on the project resembles a continuous flow. So, for example, the work within an increment will continue until the team feels they have added a significant value to the final result. It is then when they will finish the Sprint. Below, you can find a few other aspects characterizing Kanban below:

- it is usually managed by using a Kanban board, which helps to see how tasks move through the workflow
- planning happens on demand – there is no precise planning routine
- estimations of time are optional – usually, the team starts the new task from the *to-do list* when there is space for it within the items on *in progress* list
- the number of tasks labeled as *in progress* is limited
- changes to the work scope are added as needed
- roles are assigned as needed
- there are no meetings required
- the ownership depends on the defined roles and the necessities.

What is Scrumban?

As you can expect, Scrumban is the merger of Scrum and Kanban principles. It is somewhere in between the rigid Scrum framework and overly-flexible Kanban structure. What does it look like then? In Scrumban:

- the Scrum approach is used to plan, allocate and prioritize tasks
- the Kanban board is used to better visualize work and progress
- planning happens when it is needed
- estimations of time are optional
- changes to the work scope are added as needed
- the roles are as follows: Scrum Master, Product Owner, Development Team

- daily Scrum meetings are conducted; other Scrum meetings are usually added as well – depending on the need
- ownership belongs to the Product Owner but can also depend on the defined roles.

How to define which framework/tool should i use then?

Knowing that there is no clear answer to the question “which framework is the best” we have to ask yourself a different one “Which framework / tool suits best for my product ?

Important similarities of Scrum, ScrumBan & KanBan:

- relatively easy to implement but hard to master due to perfection as an end goal;
- they focus on reducing waste, continuous value delivery and creating a feedback loop;
- demand developers team to be cross functional ;
- demand proper organizational culture to bring any benefits;

What may be tricky during choosing proper framework/tool:

- some aspects of each framework/tool could bring either a benefit or a harm to your product development process so choose wisely;
- be careful with implementing top-down one framework/tool to all of the processes connected to product management (support department may need a different approach then R&D) .
- If your product is big and complicated it’s almost always better to fit a certain framework/tool bottom-up; ;
- always gather feedback from developers about direction of ongoing changes during agile transformation;
- remember that the character of your product can change in time so no solution will last forever;

SAILINGBYTE

Differences and probable best applications

Scrum	Scrum-Ban	KanBan
Framework	Framework	Tool (may be used as a part of any framework or independently)
Defines roles, responsibilities, artifacts & events	Defines roles, responsibilities, artifacts & events but you can change/cut them if they are not bringing enough value	Fitting to current workflow (no changes in roles, events etc) – just KanBan board
Time boxed	Don’t have to be time boxed	Not time boxed
Not imposing pull-system & WiP limits	Imposing pull-system & WiP limits	Imposing pull-system & WiP limits
	Probable Best Applications	
Development and maintenance of products that benefit from **empiricism** (it’s no sense to use those frameworks if we know exactly what would be an outcome of our work and/or we are not able to change product after feedback from end-customer)	Can be used **either** as a tool to a) develop products that benefit from empiricism (software development) b) purely erase waste & defects (car production etc.) c) purely smooth flow of work items by visualizing & resolving bottlenecks (maintenance & support teams)	
	Development of products that benefit most from continuous delivery (no clear deadlines, release when it’s ready)	

Development of products that are in chaotic state due to frequent changes of client's requirements and lack of clear definitions of responsibilities	Products that are currently in development stagnation due to over-analyzing future potential work items and not focusing on value delivery	Products that were developed using Agile frameworks with growing understanding of LEAN concept in whole team/organization
Development of products with no history of using agile frameworks	Development of products with some history of using Scrum framework	Products that were developed using Agile frameworks with growing understanding of LEAN concept in whole team/organization
Development of products where feedback from end customer should be transformed into product increment not faster than after 1 sprint	Development of products where feedback from end customer should be transformed into product increment not faster than after 1 sprint	Development of products where feedback from end customer have to transform into product increment in quickest time period possible

Scrum vs Agile vs Kanban vs Scrumban – which should you use?

As you can probably suspect, there is no perfect framework. At the same time, there is no bad framework. The one you should choose in your project management depends on your circumstances. Therefore, I prepared a short guide on the types of situations where it is best to use a specific framework:

- Use Scrum when your project needs a quick initial delivery and swift progression (large, long-term projects)
- Use Kanban in continuous product manufacturing, for example, while providing service, support, maintenance, bug fixing
- Use Scrumban with complex projects which have product and support features (startups, fast-paced projects)

However, while choosing the framework, you should also consider the Agile team members. If your team members do not like a strict framework, their performance might suffer if you implement the Scrum approach. The same might happen when you choose to use the Kanban approach within a team that does not work well in a

non-restrictive framework. In those cases, it is best to consider using Scrumban and combining the elements of Scrum and Kanban according to your team’s preference. Agile prioritizes the project result but also the people. The optimal result will not be achieved, if the team does not work well.

Sailing Byte masters Agile transformation and delivers your project in no time!

Sailing Byte believes that choosing Agile is the best approach to product development. We do not limit ourselves to Scrum or Kanban but embrace the whole spectrum that Agile transformation brings. After all, Sailing Byte and Agile methodologies have a common goal, which is delivering the best possible product to our clients. That is why, whenever the project needs a Scrum approach, we use it. Whenever Kanban is a better method, we will use it. Finally, if Scrumban allows for optimal product development, you can be sure Sailing Byte will use it. All that to bring your vision to life in the best possible version. Book a call today to discuss which Agile methodology we will use in the development of your software.