Introduction to Agile

Brought to you by Assemble You.

It's time to work on YOU. So sit back and listen to practical, actionable advice to accelerate your progress.

Today we explain what Agile is, how it differs from other project management approaches, and why Agile suits some projects more than others. If you've never heard the term before, Agile is an approach to project management that uses an iterative, incremental process.

For many years, project management has been dominated by the 'Waterfall' method, which focused on developing a detailed plan of action before commencing the actual development phase. With a Waterfall project, the end result is not delivered until the very end of the process.

A perfectly fine approach that works for a lot of projects, but it's also restrictive and demanding for managers. And it doesn't really lend itself well to dealing with changes to a project from external forces.

In a study of the IT sector's general transition from Waterfall to Agile for the *Journal of International Studies*, it was described like this:

> *The main issue with the Waterfall model is that it cannot ensure quick changes of stakeholders' requirements until the project is finished or nearly finished, thus, being more appropriate for projects which are considered to have stable or unchanged requirements for a longer period of time.*

Agile takes a different outlook, focusing on breaking the project into smaller timeboxes and delivering something of value at the end of each one, which can then be analyzed and critiqued while the next timebox is already underway.

The origins of Agile can be traced back to the 1950s, but the term wasn't popularized until the 21st Century. It was in 2001 when a group of software developers met to

discuss alternative, lightweight development methods and together published *The Manifesto for Agile Software Development*.

This publication outlined the twelve core principles and clarified that Agile was never intended to abolish the compelling aspects of traditional approaches. It was intended to adapt them to the ever-changing software development industry.

When introducing the manifesto, developer Jim Highsmith said this:

> *We embrace modeling, but not in order to file some diagram in a dusty corporate repository. We embrace documentation, but not hundreds of pages of never-maintained and rarely-used tomes. We plan, but recognize the limits of planning in a turbulent environment.*

The goal was to make projects more manageable and adaptable and allow for greater transparency with stakeholders.

Let's discuss applying the Agile approach to a project, and how it will lead to satisfying results:

To understand why Agile often leads to success, it's important to understand why Waterfall often leads to failure.

As we mentioned before, using the Waterfall approach to your project means doing a ton of research and analysis ahead of time and then formulating a comprehensive plan of action that everyone must follow, to the letter. Is this a bad way to manage a project? Of course not. In theory, it's a perfectly logical approach.

Extensive planning can give you a roadmap to success, and it can allow you to hit the ground running on day one. You can start the development process with confidence in your end goal and how you are going to reach it. The idea behind this is that effective planning will allow you to pick up on any major issues that might hinder the project before you actually get started and then you can plan ahead for them.

But this isn't really true, because it's impossible to know what issues your project might face. In addition, using this method turns the project into a single monumental task and that can be overwhelming. You also have nothing to show until the project is completed, leaving stakeholders out of the loop on the process.

Mark Balbes from TechBeacon illustrates this with the analogy of a Healthcare company developing a new software product over the course of nineteen months. Their Waterfall approach dictates that the first eleven months will be spent researching, gathering requirements, and designing the end result.

This is the planning phase. By the end of it, there is less than half of the overall project timeline left and not a single line of code written.

It might seem like progress is being made to those conducting the research, but that's worth nothing to the stakeholders. All they care about is the completed product, and it basically hasn't even been started yet. And because this approach necessitates the planning phase, any changes in regulation, scheduling or budget will require a new plan.

This only serves to slow things down and put the project behind schedule. These are the exact issues that the Agile approach aims to combat.

In an analysis of the methodology that was published on *HBR*, several ideas were observed as being typical of Agile projects. The article says:

> *It comes in several varieties: They include scrum, which emphasizes creative and adaptive teamwork in solving complex problems,* lean development*, which focuses on the continual elimination of waste; and* kanban*, which concentrates on reducing lead times and the amount of work in progress.*

Agile achieves these goals by breaking down the project into **small, manageable chunks that are known as 'iterations'.** The work on an individual iteration results in the completion of a part of the project that can then be delivered to the stakeholders for their consideration. In the case of software, each iteration might produce a single, functional feature.

Let's say, for example, that the software being developed is a graphic design software similar to Adobe Photoshop or Affinity Designer. Good graphic design software will have features such as a user interface, an image database, a variety of templates, design tools, and several more.

In completing this piece of software by way of Agile, each of these features may be developed in its own iteration. Once completed, they can be delivered, giving the stakeholders an opportunity to actually see the project's progress while work can begin on the next iteration.

With Waterfall, much of the time, teams work independently of each other. There will be a team dedicated to planning, one working on design, one on code, and one to test the project when it's completed. Each of these will work on the project at a different time and in different circles.

That's not how Agile works. The teams are cross-functional instead. Everyone is working together and communicating with each other at all times. So in addition to

being adaptable, transparent, and timely, Agile also enables face-to-face communication between team members.

The effect that an Agile work environment has on motivation was analyzed by Giovanni Asproni for *Agile Times* and he had this to say:

> *Agile development puts an emphasis on face-to-face communication — arguably the most effective communication channel between human beings; the team members tend to be located close to each other so the speed of communication is optimized; the customer is encouraged to interact closely with the developers so the feedback loop is shortened and the goal remains visible and clear.*

For a large-scale project of any kind, whether it's software, engineering, construction, or anything else, communication is one of the key components for success. Everybody can work in sync and any issues one specialist might face that require the assistance of another specialist can be easily dealt with through these more manageable communication channels.

So the upsides of this approach are numerous, and while it was envisioned initially *by* software developers as a solution *for* software developers, it can be applied to so many different industries.

In an analysis of Agile's effectiveness, McKinsey Quarterly reported that a leading bank underwent an agile transformation that resulted in 'doubled sales volumes, tripled high-value customer interactions and an immediate 10-20 % reduction in branches'. The numbers don't lie.

To sum up, Waterfall is and always has been a solid way to manage projects, but for many projects, it's just not reliable because it lacks a certain adaptability. This is where the Agile approach comes in.

Agile is all based around iterations, taking a big project and breaking it down into short timeboxes. It's about encouraging stakeholders' interest by allowing them consistent access to the project. And it's about enabling face-to-face communication within a team so that they can genuinely work together to achieve the best possible results.

And so for you, here's the big takeaway: Different management approaches will work for different projects, but the reality is that there are always going to be external forces changing the trajectory of what you've got planned. If you want to make sure that this doesn't seriously hinder your progress, Agile is a great choice.

Thanks for listening.

Reading List

- **[Article]** History: The Agile Manifesto, Jim Highsmith
- **[Article]** How Waterfall Development Can Fail: A Real World Scenario, Mark Balbes
- **[Article]** Embracing Agile, Darrell Rigby, Jeff Sutherland & Hirotaka Takeuchi
- **[Article]** Motivation, Teamwork, and Agile Development, Giovanni Asproni
- **[Article]** How Agile Can Power Frontline Excellence, Quentin Jadoul, Deepak Mahadevan, & Philippine Risch
- **[Article]** From Waterfall to Agile Software: Developments in the IT Sector, Alina Mihaela Dima & Maria Alexandra Maassen