Agile Software Development Defined and Applied

In this document, Agile as a concept, its development, and business world application will be defined and explained. According to Agile Alliance, "Agile Software Development is an umbrella term for a set of frameworks and practices based on the values and principles expressed in the Manifesto for Agile Software Development and the 12 Principles behind it."

Agile users and the industry tends to focus the Agile Software Development community on collaboration, meaning that teams must possess the ability to resolve issues when they arise without referencing their managers or leaders for contribution.

It also means that teams are cross-functional. Teams do not have to have specific roles, just have required skill sets to perform and execute.

Historically, Agile is thought to have emerged in 2001 when the term Agile Software Development was coined. The developers began to author and implement noteworthy methodologies to enhance and streamline performance and production. These methodologies placed focus on having close collaboration between the development team and business stakeholders. This allowed for frequent delivery of business value: tight, self-organizing teams that can mobilize ingenious ways to craft, confirm, and deliver code.

After having applied such advanced business models, the initial developers packaged the production-business model for the industry. Thus system tools such as Scrum, Extreme Programming, Feature-Driven Development (FDD), and Dynamic Systems Development Method (DSDM) came into fruition. Naturally, many other developers would mimic the idea of Agile, thus making it an industry standard.

The concept of the Manifesto for Agile Software Development was ushered in. The two primary objectives of the Agile Manifesto was: create a framework of valuable statements that form the Agile Software Development foundation and to coin the term Agile Software Development. A few moments later, the 12 Principles accompanied the idea of Agile Software Development, of which the origin site can be reached here.

Over time, as Agile became more widely known, a sort of ledger of developers who performed Agile Software Development formed. This included the people and organizations who helped them via consulting, training, frameworks, tools, etc.

As far as resources, Agile Alliance continues to cultivate resources to help end users adopt Agile practices and improve one's ability to develop software with agility. The Agile Alliance website provides access to those resources such as videos and presentations from conferences, experience reports, an Agile Glossary, a directory of community groups, and several other applicable resources.

Most Agile implementers recognize Agile to be a behaviorism, an ethos, or a mentality so to speak. Agile is fueled by the Agile Manifesto's values and objectives. Those values and objectives provide a framework on how to create and respond to deviations from the initial plan and how to deal with random uncertainty.

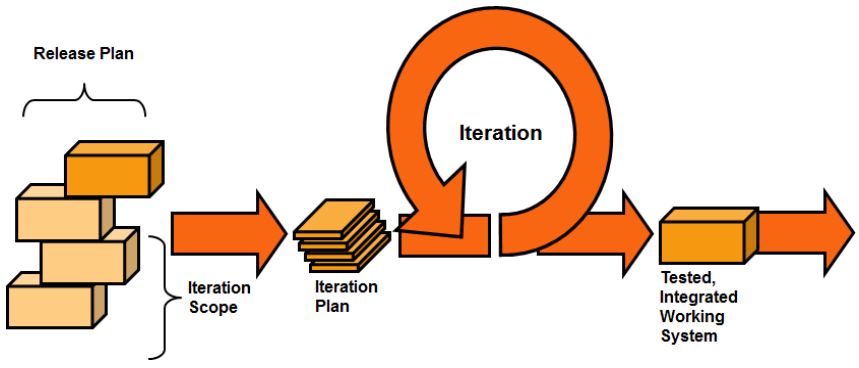
***Note**: A Methodology is the set of conventions that a team agrees to follow. Agile becomes an adjective once it is applied as an ethos. It describes how one can perform activity.*

Industry business agility requires a *recognition* in order for people in an organization to operate with an Agile mindset. This means the organization must apply Agility at all levels and support the Agile methodology. Agile Software Development reached its apex when organizations adapted its structure and operations to work in an uncertain environment in which Agile Software Development could flourish and reign, giving way to The Age of Agile.

Agile Step by Step Explained:

According to Agile in a Nutshell, Agile is defined as: "a time-boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end."

The tool system ultimately works by compartmentalizing projects into bits of user functionalities called user stories, prioritizing them, and then continuously delivering them in short two week cycles called iterations.



Agile users would proceed to create a list of client requested features the client would like to see in their software. This data and process is also known as user stories as well. Once established, the user stories become the To Do list of the project.

Once the To Do List is composed, using Agile estimation techniques, implementers size the stories relatively to each other. Agile implementers will then estimate the time needed to devote to each user story. Members on the projects will need to consult with the customer to prioritize their list, providing what is Primary, Secondary, Tertiary, so on and so forth.

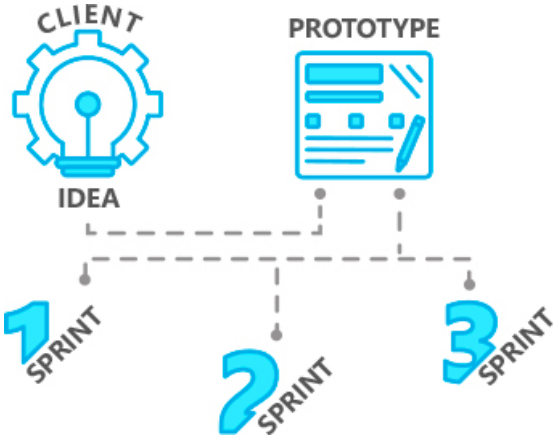
According to Agile in a Nutshell: “An Agile iteration is a short one to two week period where a team takes a couple of their customers' most important user stories and builds them completely as running-tested-software.”

This is the stage of analysis, design, coding, testing. Agilist working this way, they can deliver updates and results every couple weeks. This can yield some demo or beta working software. This method also facilitates a great way to track progress.

Once the foundation services are established, Agile implementers can begin to build, iterating, and referencing any feedback from the client with the project timeline. Once the project has considerable contributions, there may emerge two developing results. The project is advancing with no issues, or an issue has arrived that will impact the service or project debut. The team can either: a) do less and cut scope, which is often recommended or b) push out the date and ask for more money to the budget to patch up the issue(s) and keep building.

In the industry of projects, software, and services an ***Iterative Development*** means starting with the simple, and adding to it incrementally over time. This means evolving the structure-framework, accepting that requirements are going to change, and continuously refining and tweaking the product as it builds.

The ability to modify as the project grows is known as ***Adaptive Planning***, or updating and adapting the project plan rather than shutting it down or eliminating from the roster all together.



Typically, this is done by ***Flex on Scope***, meaning Aglists have many skills and apply them with the business needs, as opposed to focusing on their premier skill set or the originally assigned section of the project. Agile implementers must be able to adjust time, budget, and quality, and apply flexing around scope in order to maintain the integrity of the project.

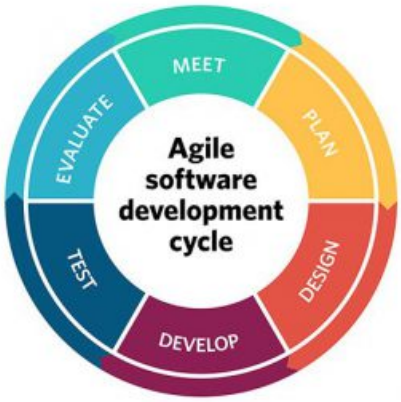
Agile Software Development seeks to challenge the notion that the cost of change can be high and volatile, providing solutions that can be relatively flat. Agilist achieve this through a combination of modern software engineering practices and open and honest planning.

According to Agile in a Nutshell, Agile can quantified in five fixed stages:

Quality improves because testing starts from day one
Visibility improves because you are 1/2 way through the project when you have built 1/2 the features
Risk is reduced because you are getting feedback early
Customers are happy because they can make changes without paying exorbitant costs

User stories are features the client may one day like to see in their software. In true Agile form, user stories are the basis of the Agile project. Aglists must appropriately size and prioritize these stories like any other client list of requests.

This data is obtained by gathering clients into workshops, using tools like: flowcharts, screens, storyboards, mockups, etc and then proceed to break the functionality down into simple words and phrases clients understand.



The ***Style of Estimation***, known as relative over absolute, forms the cornerstone of Agile Planning. By sizing the stories relatively, and feeding real time data back into the plan, the team can make some accurate predictions about the future based on past project development.

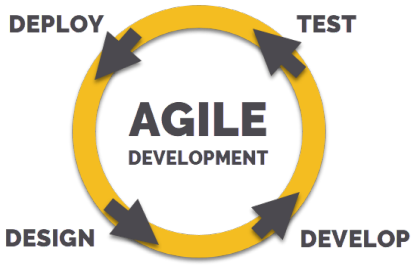
Some honorable mentions of premier Agile Software Development users are Riot Games and Spotify. Observationally, these firms grew exponentially and continue to be run according to Agile principles and values.

Agile Software Development expanded beyond individual software development teams into a traditional hierarchical bureaucracy in order to achieve efficient, reliable management at scale.

This business type circumnavigates the power trickling down from the top as a prototypical business model. Agile Software Development accommodates the future of a firm depends on Agilists doing the work to accelerate project innovation and add value to clients.

According to Forbes, “Agile Software Development values transparency and continuous improvement ahead of predictability and efficiency, it recognizes that open interactive conversations are more valuable than top-down directives, it stops doing anything that is not adding value to the clientele, it realizes that the key to success is not to do more work faster; the key is to be smarter by generating more value from less work and delivering it sooner.”

Ultimately, Agile Software Development demotes the traditional management distinction between exploitation and exploration as a business model. Agile incorporates all parts of the organization, that Agilists are continuously expanding on how to add more value to the project and end user clientele. Agile Software Development results in favorable returns to the organization in of itself.



Tags: #agile, #agilesoftwaredevelopment, #softwaredevelopment, #development, #software, #deployment, #project, #projectplanning, #userstories, #iterations, #irativedevelopment, #testing, #agiletesting, #styleofestimation, #estimation, #flexonscope, #flex, #scope

Resources:

Agile Alliance
Agile in A Nutshell
Forbes Agile