scale agile

Agile Model: What Is It And How Do You Use It?

January 16, 2017 | Andrew Powell-Morse | SDLC

Back in February of 2001, a small group of seventeen software developers met in the high elevations of Snowbird, Utah to discuss software development methodologies. Out of this assemblage emerged the Manifesto for Agile Software Development, a publication that outlined the group's vision for lightweight software development, and which would go on to dramatically shape the landscape of modern software development for years to come, up to the present day.

From the four fundamental values this group of developers set forth emerged one of the most widely adopted software development methodologies in modern history – agile model. At its core, the agile model emphasizes the need for every project to be handled differently, based on the individual needs of the project, the schedule, and the team behind it. Unlike other SDLC models, Agile focuses less on specific requirements or guidelines, and far more on abstraction of these best practices to allow for greater flexibility, or *agility*, during the development process.

Throughout this article we'll dive deep into what the agile model is, the values and principles that make up the core structure, and the overall advantages and disadvantages of utilizing it during modern software development life cycles.

Some more specific takes on SDLC include:

Rapid Application Development	Test-Driven Development	Waterfall Model
Iterative Model	Extreme Programming	Scaled Agile Framework
Software Development Life Cycle	Scrum	Rational Unified Process
Big Bang Model	V-Model	Conceptual Model
Kaizen Model	Kanban Model	Spiral Model

The Agile Values

Within the original Manifesto for Agile Software Development the authors focused on four fundamental, core values that underpin solid software development.

- Individuals and interactions: Rather than solely emphasizing systems and tools, the focus should be on the people within the team and the interactions they have while working together on the project. For a project to be successful, it should adapt to not just the systems or budget available, but most importantly to the people working on it. If team members are efficient and effective at working together, the end result will be a polished and optimized product.
- Working software: While documentation can certainly be very beneficial during development, it is far better to produce a working product, or even a simple prototype, that illustrates the design goals or the components used throughout the application. This is beneficial not only to other team members working on development, management, and marketing, but especially to clients or testers who would otherwise be forced to rely on a handful of documents and photoshopped illustrations to understand how the application is expected to function.
- Customer collaboration: As with Rapid Application Development, or any other development model born from the roots of the agile model, it is critical that the project be constantly open, willing, and able to respond to customer feedback and behavior. By keeping customers or clients in the loop throughout the entire life cycle, everyone involved will be on the same page and there will not be any surprises at the end or massive rewrites necessary because a module or integration wasn't clear for all parties involved.
- Responding to change: Perhaps the most critical principle across the entirety of the agile model is the ability for the project to adapt and respond to the ever-changing needs of everyone and everything involved. As development progresses, software technologies will change, the team will shift, clients will hem and haw, and throughout it all, the project should remain malleable and remain capable of adapting along with these needs.

The Agile Principles

As further outlined in the Manifesto, the agile model is described using twelve key principles, atop which the development life cycle should take place.

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.
8. Agile processes promote sustainable development. 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.

Pros of the agile model

- Emphasis of Modern Techniques: While a core value of the agile model places emphasis on people over technologies, stepping outside the realm of technologies themselves and into a pure focus on techniques brings about powerful, agile practices such as test-driven development, automated unit testing, refactoring, and iterative development.
- Highly Adaptive: As one of the fundamental agile values states, a key component to the agile model, and which partially makes it such a good launching pad for the entire software development life cycle, is the capability of the project to rapidly adapt to any necessary changes. Whether this is from rapid iterations informing the changing needs within the code or client feedback forcing a reshaping of the sign-up procedures, a properly Agile project is able to quickly and effectively change course as needed.
- Constant Customer Feedback: Although the constant communication with customers and clients requires a dedicated team member to take on this responsibility, it is a critical component to ensuring the success of the product from both a development perspective as well as from that of the client.
- Allows for Iterative Development: Common models like the Iterative Model are based on the fundamentals of the agile model, and for good reason: It allows the project to be started with relatively little upfront planning or cost. From these early components the project can evolve over time as new incremental iterations are made, constantly learning from past iterations and improving on them for future updates.

Cons of the agile model

- Potential for Increased Technical Debt: The concept of technical debt describes the act of implementing a solution that is easier to complete right now, in favor of the solution that may be better overall in the long run. Within the scope of the agile model, it is not uncommon for technical debt to begin to run rampant, as rapid development techniques and frequent iterations often mean developers feel hurried and thus forced to implement faster but generally less-than-ideal band-aids. This issue can largely be reduced by ensuring the team properly integrates refactoring, pair programming, and other techniques which emphasize collaboration amongst team members.
- Difficult to Make Additions Within an Iteration: Often referred to as *use cases* in other development models, a story in the agile model simply refers to a description of some new project requirements. When actively utilizing the agile model for an ongoing project, it can sometimes be difficult to implement a new story into the current iteration, since it forces backtracking and heavy mental overhead on how to implement the new requirements of this story into the existing iteration that has largely already being developed. In such cases, it is often necessary to delay the implementation of the new story until the next iteration comes about.
- Minimal Emphasis on Documentation: Unlike more traditional models like the Waterfall Model, the agile model largely forgoes initial efforts to heavily design and document the project requirements or scope, in favor of getting into the meat of the project and beginning that iterative process. This can be a challenge for some projects, particularly within development teams which may not be accustomed to this style of agile development and which may have more traditional experience instead.