# Iterative Model: What Is It And When Should You Use It?

[December 15, 2016](https://airbrake.io/blog/sdlc/iterative-model) [Andrew Powell-Morse](https://airbrake.io/blog/author/andrewpomo) in [SDLC](https://airbrake.io/blog/category/sdlc)

The iterative model is a particular implementation of a software development life cycle (SDLC) that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete. When discussing the iterative method, the concept of incremental development will also often be used liberally and interchangeably, which describes the incremental alterations made during the design and implementation of each new iteration.

Throughout this article we will take a deeper dive into the iterative model, including where it originated from, how the process is actually implemented within a project, and both the advantages and disadvantages that can arise from its use, so let’s get crackin’!

Some more specific takes on SDLC include:

The Process

Unlike the more traditional waterfall model, which focuses on a stringent step-by-step process of development stages, the iterative model is best thought of as a cyclical process. After an initial planning phase, a small handful of stages are repeated over and over, with each completion of the cycle incrementally improving and iterating on the software. Enhancements can quickly be recognized and implemented throughout each iteration, allowing the next iteration to be at least marginally better than the last.

* **Planning & Requirements**: As with most any development project, the first step is go through an initial planning stage to map out the specification documents, establish software or hardware requirements, and generally prepare for the upcoming stages of the cycle.
* **Analysis & Design**: Once planning is complete, an analysis is performed to nail down the appropriate business logic, database models, and the like that will be required at this stage in the project. The design stage also occurs here, establishing any technical requirements (languages, data layers, services, etc) that will be utilized in order to meet the needs of the analysis stage.
* **Implementation**: With the planning and analysis out of the way, the actual implementation and coding process can now begin. All planning, specification, and design docs up to this point are coded and implemented into this initial iteration of the project.
* **Testing**: Once this current build iteration has been coded and implemented, the next step is to go through a series of testing procedures to identify and locate any potential bugs or issues that have have cropped up.
* **Evaluation**: Once all prior stages have been completed, it is time for a thorough evaluation of development up to this stage. This allows the entire team, as well as clients or other outside parties, to examine where the project is at, where it needs to be, what can or should change, and so on.

Now the real fun begins! This is the crux of the entire iterative model, whereby the most recently built iteration of the software, as well as all feedback from the evaluation process, is brought back to the planning &amp; development stage at the top of the list, and the process repeats itself all over again.

Advantages of the Iterative Model

* **Inherent Versioning**: It is rather obvious that most software development life cycles will include some form of versioning, indicating the release stage of the software at any particular stage. However, the iterative model makes this even easier by ensuring that newer iterations are incrementally improved versions of previous iterations. Moreover, in the event that a new iteration fundamentally breaks a system in a catastrophic manner, a previous iteration can quickly and easily be implemented or “rolled back,” with minimal losses; a particular boon for post-release maintenance or web applications.
* **Rapid Turnaround**: While it may seem like each stage of the iterative process isn’t all that different from the stages of a more traditional model like the waterfall method — and thus the process will take a great deal of time — the beauty of the iterative process is that each stage can effectively be slimmed down into smaller and smaller time frames; whatever is necessary to suit the needs of the project or organization. While the initial run through of all stages may take some time, each subsequent iteration will be faster and faster, lending itself to that agile moniker so very well, and allowing the life cycle of each new iteration to be trimmed down to a matter of days or even hours in some cases.
* **Suited for Agile Organizations**: While a step-by-step process like the waterfall model may work well for large organizations with hundreds of team members, the iterative model really starts to shine when its in the hands of a smaller, more agile team. Particularly when combined with the power of modern version control systems, a full “iteration process” can effectively be performed by a number of individual team members, from planning and design through to implementation and testing, with little to no need for outside feedback or assistance.
* **Easy Adaptability**: Hinging on the core strength of constant, frequent iterations coming out on a regular basis, another primary advantage of the iterative model is the ability to rapidly adapt to the ever-changing needs of both the project or the whims of the client. Even fundamental changes to the underlying code structure or implementations (such as a new database system or service implementation) can typically be made within a minimal time frame and at a reasonable cost, because any detrimental changes can be recognized and reverted within a short time frame back to a previous iteration.

Disadvantages of the Iterative Model

* **Costly Late-Stage Issues**: While not necessarily a problem for all projects, due to the minimal initial planning before coding and implementation begin, when utilizing an iterative model, it is possible that an unforeseen issue in design or underlying system architecture will arise late into the project. Resolving this could have potentially devastating effects on the time frame and costs of the project as a whole, requiring a great deal of future iterations just to resolve one issue.
* **Increased Pressure on User Engagement**: Unlike the waterfall model, which emphasizes nearly all user/client engagement within the initial stages of the project during a brief crunch time period, the iterative model often requires user engagement throughout the entirety of the process. This is sometimes an unfortunate obligation, since each new iteration will likely require testing and feedback from users in order to properly evaluate any necessary changes.
* **Feature Creep**: Not only does the iterative model require user feedback throughout the process, but this also inherently means the project may be subject to undesired feature creep, whereby users experience the changes in each iteration, and are inclined to constantly put forth new requests for additional features to be added to future versions.

With the adept agility and rapid turnaround of iterative model projects, you need an exception management tool that can keep up the pace, *without* the overhead of user-intervention. [Airbrake’s error monitoring software](https://airbrake.io/account/new?utm_source=blog&utm_medium=end-post&utm_campaign=airbrake-what-is-iterative-model&utm_content=Iterative%20Model:%20What%20Is%20It%20And%20When%20Should%20You%20Use%20It?%20|%20Airbrake%20Blog) provides real-time error monitoring and automatic exception reporting for all your development projects. Airbrake’s state of the art web dashboard ensures you receive round-the-clock status updates on your application’s health and error rates. No matter what you’re working on, Airbrake easily integrates with all the most popular languages and frameworks. Plus, Airbrake makes it easy to customize exception parameters, while giving you complete control of the active error filter system, so you only gather the errors that matter most.

Check out [Airbrake’s error monitoring software](https://airbrake.io/account/new?utm_source=blog&utm_medium=end-post&utm_campaign=airbrake-what-is-iterative-model&utm_content=Iterative%20Model:%20What%20Is%20It%20And%20When%20Should%20You%20Use%20It?%20|%20Airbrake%20Blog) today and see for yourself why so many of the world’s best engineering teams use Airbrake to revolutionize their exception handling practices!