What is Agile Development?

10/11/2017 • 4 minutes to read

In this article

Diligent backlog refinement
Integrate early and often
Minimize technical debt
Always Be Agile

By: Dan Hellem

Agile development is a term used to describe iterative software development. Iterative software development shortens the software development lifecycle. Agile development teams execute the entire software development lifecycle in smaller increments, usually called sprints. Sprints are typically 1-4 weeks long. Agile development is often contrasted with traditional or waterfall development, where larger projects are planned up front and executed against that plan.

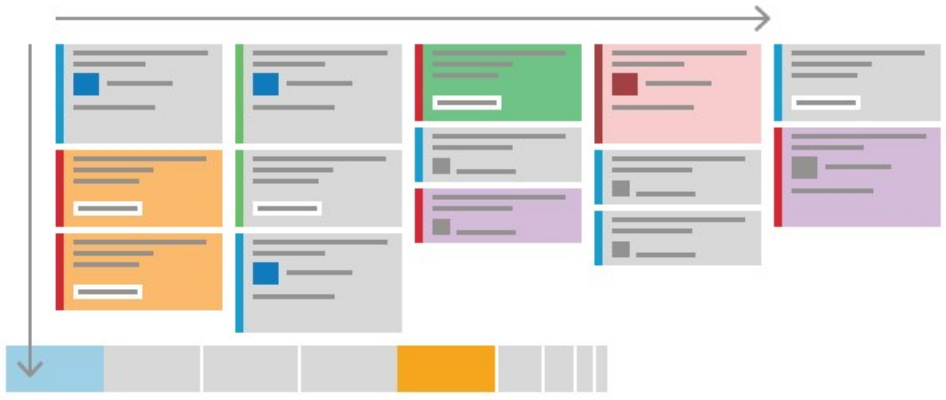
Delivering production quality code every sprint requires the agile development team to account for the accelerated pace. All coding, testing, and quality verification must be done each and every sprint. Unless a team is properly set up, it can fail. And sometimes fail miserably.

This article lays out a few key success factors for agile development teams:

- Diligent backlog refinement
- Integrate early and often
- Minimize technical debt

Diligent backlog refinement

An agile development team works off of a backlog of requirements, often called “user stories”. The backlog is prioritized so the most important user stories are at the top. The product owner owns the backlog and adds, changes, and reprioritizes user stories based on the customer’s needs.



One of the biggest drags on an agile team’s productivity is a poorly defined backlog. A team cannot be expected to consistently deliver high quality software each sprint unless they have clearly defined requirements.

The product owner's job is to ensure that every sprint, the engineers have clearly defined user stories to work with. The user stories at the top of the backlog should always be ready for the team to execute on. This is called backlog refinement. Keeping a backlog ready for an agile development team takes an incredible amount of effort and discipline.

When refining the backlog, remember the following:

Refining user stories is often a long-lead activity

Elegant user interfaces, beautiful screen designs, and customer delighting solutions all take time and energy to create. Diligent product owners refine user stories 2-3 sprints in advance. They account for design iterations and customer reviews. They work to ensure every user story is something the agile team is proud to deliver to the customer.

A user story is not refined unless the team says it is

The team needs to review the user story, and agree it's ready to work on. If a team has not seen the user story until day 1 of a sprint, that's a big red flag.

User stories further down the backlog can remain ambiguous

Don't waste time refining lower priority items. Stay intently focused on the top of the backlog.

Integrate early and often

Continuous integration and continuous delivery (CICD) sets your team up for the fast pace of Agile Development. As soon as possible, automate your build, test and deployment pipeline. It should be one of the first things you complete when starting a new project.

With automation, your team will avoid slow, error prone, and time-intensive manual deployment processes. And let's face it, if you are releasing every sprint, your team doesn't have time to do this manually.

CICD also influences your software architecture. They ensure your delivers buildable and deployable software. When you implement a difficult-to-deploy feature, you become aware immediately as the build and deployments will fail. CICD forces you to fix deployment issues as they occur, ensuring your product is ever ready to ship.

CICD key activities are:

Unit testing

Unit tests are your first defense against human error. Unit tests should be considered part of coding and checked in with the code. Executing unit tests should be considered part of your build. Failed unit tests mean a failed build.

Build automation

Have your build system automatically pull code and tests directly from source control when builds execute.

Branch and build policies

Configure your branch and build policies to build automatically as your team checks code into a specific

branch.

Deploy to an environment

Setup a release pipeline that automatically deploys your built project(s) to an environment that mimics production.

Minimize technical debt

With personal finances, it's easier to stay out of debt than to dig out from under it. The same rule applies for technical debt. Technical debt includes anything the team must do to deploy production quality code and keep it running in production. Examples are bugs, performance issues, operational issues, accessibility, and others.

Keeping on top of technical debt requires courage. There are many pressures to delay fixing bugs. It feels good to work on features and ignore debt. Don't be fooled. Sooner or later, somebody must pay the technical debt. Just like financial debt, technical debt becomes harder to pay off the longer it exists. A smart product owner works with their team to ensure there is time to pay off technical debt in every sprint. Balancing technical debt reduction with feature development is a difficult task. Read Creating productive, customer focused teams for ideas on how to manage this.

Always Be Agile

Being agile means learning from what you do and continually improving. Agile development provides many more learning cycles than traditional project planning. Each sprint provides something new for the team to learn.

For example:

- A team delivers value to the customer, gets feedback, and then modifies their backlog based on that feedback.
- They learn that their automated builds are missing key tests and include work in their next sprint to address it.
- They find that certain features perform poorly in production and make plans to improve performance.
- Someone on the team hears of a new practice and the team decides to try it out for a few sprints.

If you are starting with agile development, expect more learning opportunities. Way more. An Agile team doesn't waste those opportunities.

Get started with free agile tools in Azure Boards.

Dan is a Senior Program Manager with Microsoft's Azure DevOps. Dan focuses on the Agile space as well as the customer adoption of the service. Before coming to Microsoft in 2012 Dan spent his career building applications using Microsoft technologies and assembling Agile teams centered on delivering high quality software to customers.