Agile release planning: How to effectively plan for success

very project needs a plan. But sometimes, even the best, most carefully drafted plans need to change.

That's the basic premise behind agile project management, a project management methodology that's wildly popular among software development teams. In 2021, 86% of software development teams were using agile, and it's quickly becoming the dominant practice in industries where iteration and flexibility are essential.

Most agile software development teams take an iterative, release-based approach to product development, releasing more frequent updates with fewer features in each. Building project schedules to match this model requires a different approach, called agile release planning.

In this article, we'll walk you through the basics of agile release planning: what it is, why it matters, how to build an agile release plan, and best practices to keep in mind as you go.

What is agile release planning?

Agile release planning (sometimes called Scrum release planning) is where teams build and release products in iterations or incremental stages rather than all at once or grouped into distinct major releases.

It is most common in software product development, but it can be helpful in other disciplines where iterative, flexible releases make sense (website project management is a related example, and even marketing project planning can benefit from some of the concepts).

Why is agile release planning important?

One reason is that businesses are increasingly going agile, and they need a release plan that works in that framework. A 2020 McKinsey study finds that businesses that go agile generally:

- Increase operational performance by 30% to 50%
- Increase employee engagement by 20% to 30%
- Improve financial performance between 20% and 30%

And another report shows that agile had three times the success rate of waterfall for most businesses.

So it's no surprise that organizations are moving in this direction. As they do, they need a release planning process that fits the agile framework.

Another purpose of release planning is that audiences (from customers to executives) can see ongoing results, enjoy ongoing product improvements, and achieve higher new feature adoption. This is a massive advantage in the competitive world of software and tech services.

4 steps for building an agile release plan

Creating a successful agile release plan requires careful planning that starts long before you begin placing release dates on a calendar. Start with these four steps to ensure your agile release plan is built to succeed.

Quick note: From this point forward we're just going to assume a software development context. If you're operating in another industry, adapt as you see fit.

1) Determine your product vision

First up is establishing a product vision (or evaluating or clarifying it if one already exists). You can't accomplish effective release planning or build the product roadmap until you know what you're trying to accomplish. Usually, the product owner, team members, stakeholders, and executives will all be involved in this process. They work together to formulate a vision, taking into account the business's strengths and goals, along with market demand.

With a clear product vision in place, you as the project manager or Scrum master can continue building the agile release plan.

2) Review your product backlog and rank the features

Every software project will have a wide range of planned and requested features — things that don't exist yet but likely will eventually. If you're using the Scrum framework, you'll already have most of these listed as product backlog items.

Here's the thing: Not all the items are equally valuable. And it's up to someone like you (likely the product manager or Scrum product owner) to rank these backlog features.

Often, teams use user stories to help rank these features, putting into the user's words what a new feature will offer them. You should also review the product vision and seek input from other stakeholders who may have differing priorities.

If you don't already have a basic roadmap for the project, now's the time to sketch one out, with at least a rough outline of release goals and dates (cross-checked against the ranked backlog items).

3) Host a Scrum release planning meeting

If you’re using Scrum, now’s the time to pull stakeholders into a Scrum release planning meeting. (If you’re using another agile framework such as Kanban, you’ll still want to do something similar.)

This is where you start building and refining the release plan itself. With the right stakeholders involved, you’ll ensure that all relevant parties or departments have a chance to weigh in and that stakeholders agree on the prioritization of deliverables and features.

Not sure what to discuss in this meeting? Start with these agenda items:

Have a product planning discussion

Now that you have the right people in the room (or on the Zoom call), discuss the product plan and roadmap. Make sure the team agrees that those foundational elements are correct, sensible for the project, and up to date.

Evaluate previous sprints architecture

Your software development team didn’t all start yesterday (at least we hope), so you’re rarely starting from nothing. Review both the architecture and the level of success from a previous set of sprints. These could be from your last project or, as the current project develops, from the previous release of the current project.

What did you learn from the previous sprints? Were any of them delayed, or do user stories from the last sprint spill into this one? Did you discover any dependencies or poor assumptions?

Take the time to learn from previous similar experiences so your current product or release is even stronger.

Determine velocity for current sprints

Velocity, or how quickly teams can realistically build out features and reach release or milestone points, is a key element of any sprint-based methodology. Determine what velocity is reasonable using data from a similar project (or, as your project develops, from a previous iteration).

Those user stories come back into play here as a part of sprint planning. Assign story points to each user story — representing how much time your agile team will need to build out a given story — and then use those points to build out sprints within a release or iteration.

Establish the release's "Definition of Done"

The whole point of agile release planning is that "done" and "final" are a little bit less important — instead, your product is continually improving. But this can create a lack of clarity over what "done" looks like in shorter-range tasks and in iterative releases.

So as you're planning a release, take the time as a group to determine what the definition of "done" is for a given release. At first, it's probably reaching MVP (minimum viable product). For future releases, it usually looks like having a stable release that contains the features you planned for it to have.

By gaining agreement on the right definition of "done" for a specific release, you'll have an easier time keeping teams focused throughout their sprints.

4) Finalize and release the planning schedule

Now that you've gathered information and attained buy-in, you're ready to finish building out your agile release plan and deploy it with the team.

Make sure you don't skip that second part: Everyone involved in the sprint must have access to your finished release plan. It — along with the guidance of the project manager or Scrum master — is what keeps teams focused on the right tasks and shows them whether they're staying on schedule.

Release planning best practices

As you work on building release plans, keep these best practices in mind.

Always start with a clear vision in mind

To arrive at the right destination at the right time, you must know where you want to go. With a clear vision of the desired outcome of a release, you'll have a much easier time building out the steps to get to that outcome.

Stick to flexible deadlines

While achieving deadlines should always be a goal, it isn't the only goal that matters. That's why your team or organization adopted agile in the first place, right?

So remember, it's okay to be flexible with deadlines when they turn out to be unrealistic. Most of the time, if you have to choose between sacrificing features or quality and being a little late on an internally imposed deadline, the choice is clear.

Stagger your product releases

Sometimes, you only have one developer or department that can do a specific part of a project. You end up having to share that person across multiple project teams.

If this is the case at your organization, then staggering the release dates for those products is ideal. Otherwise, you'll end up with parallel sprints and parallel release plans, which tend to put too much concurrent pressure on the shared resource.

With too much pressure on the only developer or department that can do a particular task on both projects, you'll create bottlenecks that threaten both projects.

By the way, balancing the workload of shared resources is infinitely easier with Teamwork.com project management software. That's true for software teams — and for the creatives and designers that support them with visual and marketing content. See how Teamwork.com helps creative and design teams track and plan projects.

Remember that planning shouldn't end at the point of release

Agile release planning is all about iterative updates, which means that every release is both an end and a beginning. The point of release is certainly something to celebrate, but make sure you're already starting to plan for the next release.

Manage projects and release planning with Teamwork.com

Agile release planning is a great way for software development teams to make steady, measurable progress and continue delivering results for customers over an entire product lifecycle. With the guidance we've provided here, you're well on your way to better release planning.

There's just one more piece of the puzzle: the project planning software and platform you'll use to manage it all. Teamwork.com is the ideal choice for product teams of all kinds, including software development teams.