It’s an exciting time to be agile! For the first time, our industry has found a real, sustainable way to solve problems that generations of software development teams have been struggling with. Agile teams use simple, straightforward practices that have been proven to work on real-life projects. But wait a minute... if agile is so great, why isn’t everyone doing it? It turns out that in the real world, a practice that works really well for one team causes serious problems for another team, and the difference is the team mindset. So get ready to change the way you think about your projects!
The new features sound great...

Meet Kate. She’s a project manager at a successful Silicon Valley startup. Her company builds software that’s used by video and music streaming services and internet radio stations to analyze audiences in real time and choose programming suggestions that make their viewers or listeners happy. And now Kate’s team has an opportunity to deliver something that will really help the company.

I’LL TALK TO THE TEAM ABOUT GETTING THOSE NEW ADVANCED AUDIENCE ANALYTICS FEATURES INTO THE NEXT RELEASE.

THANKS, KATE. IF THE TEAM CAN GET THIS DONE QUICKLY, OUR BIGGEST CLIENT WILL ADD FIFTY LICENSES, AND WE’LL ALL GET A BIG BONUS THIS YEAR!

Kate’s a project manager for a software team.

Ben is the product owner. His job is to talk to clients, figure out what they need, and come up with new features that they’ll use.
...but things don’t always go as expected

Kate’s discussion with the project team didn’t go nearly as well as she’d hoped. What’s she going to tell Ben?

Kate: So if we can get those new advanced audience analytics features into the next release, we’ll all get a big bonus.

Mike: Well, that sounds like it would be great.

Kate: Fantastic! So we can count on you guys?

Mike: Hold on! Not so fast. I said it *sounds* like it *would* be great. But there’s no way it’s happening.

Kate: Wait, what?! Don’t mess with me, Mike.

Mike: Look, if we’d known about this change four months ago when we were designing the audience data analysis service, this would be easy. But now we’d have to rip out a huge chunk of code and replace it with... well, I don’t want to get into technical details.

Kate: Good. I don’t want to hear them.

Mike: So... are we done here? Because my team’s got a lot of work to do.
Agile to the rescue!

Kate’s been reading about agile, and she thinks it might help her get those features into the next release. Agile’s gotten really popular with software teams because the ones that have “gone agile” often talk about the great results they get. The software they build is better, which makes a big difference to them and their users. Not only that, but when agile teams are effective, they have a much better time at work! Things are more relaxed, and the working environment is a lot more enjoyable.

So why has agile gotten so popular? Lots of reasons:

- When teams go agile, they find that it’s a lot easier to meet their deadlines.
- They also find that they can really cut down on bugs in their software.
- Their code is a lot easier to maintain—adding, extending, or changing their codebase is no longer a headache.
- The users are a lot happier, which always makes everyone’s lives easier.
- And best of all, when agile teams are effective, the team members’ lives are better, because they can go home at a reasonable hour and rarely have to work weekends (which, for a lot of developers, is a first!).

A daily standup is a good starting point

One of the most common agile practices that teams adopt is called the daily standup, a meeting that happens every day, during which team members talk about what they’re working on and their challenges. The meeting is kept short by making everyone stand for the duration. Adding a daily standup to their projects has brought success to a lot of teams, and it’s often the first step in going agile.
Kate tries to hold a daily standup

To Kate’s surprise, not everyone on Mike’s team shares her excitement for this new practice. In fact, one of his developers is angry that she’s even suggesting that they add a new meeting, and seems to feel insulted by the idea of attending a meeting every day where he’s asked prying questions about his day-to-day work.

The new features are really important. Let’s hold a daily status meeting so I can get updates from you and the team every day. That’s a great agile practice that we can all get behind!

We already have too many meetings! If you don’t trust us to do the job, find another team to do it.

Kate thinks Mike and his team are being irrational, but maybe they have a point. What do you think?

So what’s going on here? Is Mike being irrational? Is Kate being too demanding? Why is this simple, well-accepted practice causing a conflict?
Different team members have different attitudes

Kate ran into problems adopting agile almost as soon as she started—and she’s not alone.

The truth is that a lot of teams simply haven’t had as much success with agile as they’d hoped they would. Did you know that well over half of companies that build software have experimented with agile? Despite the success stories—and there are many of them!—a lot of teams try agile, but end up with results that they’re not particularly happy with. In fact, they feel a little ripped off! It seemed like agile came with a promise of big changes, but trying to get agile working on their own projects never seemed to work out.

And that’s what’s happening to Kate. She created a plan all on her own, and now she wants to get status updates from her team. So she’s started dragging a reluctant team to her daily standup meeting. She’s able to get them in the room. But will it really make a difference? She’s worried about how people are deviating from her plan, so she’ll concentrate on getting a status update from each person. Mike and his developers, on the other hand, want the meeting to end as quickly as possible so they can get back to “real” work.

In Kate’s less-than-effective daily standup, each person tunes out everyone else while they’re waiting to give their updates, then says as little as possible when it’s time to speak. She still gets some useful information, but at the cost of conflict and boredom—and nobody in the meeting is collaborating.

Two people can see the same practice very differently. If they don’t both feel like they’re getting something out of it, it can make the practice much less effective.

I only find out about problems when it’s too late to do something about them.

When you keep dragging us to meetings, we don’t any have time to write code.

Two people can see the same practice very differently. If they don’t both feel like they’re getting something out of it, it can make the practice much less effective.
THAT’S JUST HOW SOFTWARE PROJECTS ARE, RIGHT? THINGS THAT WORK IN TEXTBOOKS DON’T REALLY WORK OUT IN REAL LIFE. THERE’S NOTHING WE CAN DO ABOUT IT, RIGHT?

No! The right mindset makes practices more effective.

Let’s be clear: the way Kate is running her standups is how many daily standups are run. And while it’s not optimal, a daily standup that’s run this way will still produce results. Kate will find out about problems with her plan, and Mike’s team will benefit in the long run because those problems that do affect them can be taken care of sooner rather than later. The whole thing doesn’t take much time every day, and that makes it worth doing.

But there’s a big difference between an agile team that goes through the motions and one that gets great results. The key to that difference is the mindset the team brings to each project. Believe it or not, the attitude that each person takes toward a practice can make it much more effective!

The attitude that each team member brings to a practice like the daily standup makes a huge difference in how effective it is. But even when everyone tunes out, the meeting is still effective enough to make it worth doing, even if it’s boring.
A better mindset makes the practice work better

What would happen if Kate and Mike had a different mindset? What if each person on the team approached the daily standup with an entirely different attitude?

For example, what would happen if Kate felt like everyone on the team worked together to plan the project? Then she would genuinely listen to every single developer. If Kate changes her attitude toward the standup, she’ll stop trying to figure out how they’ve deviated from her plan so that she can correct them. The focus of the meeting changes for her: now it’s about understanding the plan that everyone on the team worked together to create, and her job is about helping the whole team do their work more effectively.

That’s a very different way of thinking about planning, one that Kate was never taught in any of her project management training courses. She was always taught that it was her job to come up with a project plan and basically dictate it to the team. She had tools that let her measure how well the team followed her plan, and strict processes that she would enforce to make changes to it.

Now things are totally different for her. She realized that the only way she could make the daily standup work is if she puts effort into working with the team so that everyone on the team can work together to figure out the best approach to the project. Then the daily standup turns into a way for the whole team to work together to make sure everyone is making solid decisions, and that the project is on track.

Kate used to get really frustrated when she discovered changes to her project plan, because it was usually too late for the team to effectively change course.

Now that the daily standup is in place, the whole team works with her every day to find those changes, so they can make them much earlier. That’s a lot more effective!
And what if Mike felt like this meeting wasn’t just about giving status updates, but about understanding how the project is going, and coming together every day to find ways that everyone can work better? Then the daily standup becomes important to him.

A good developer almost always has opinions not just about his own code, but about the whole direction of the project. The daily standup becomes his way to make sure that the project is run in a sensible, efficient way—and Mike knows that in the long run that will make his job of coding more rewarding for his team, because the rest of the project is being run well. And he knows that when he brings up a problem with the plan during the meeting, everyone will listen, and the project will run better because of it.

Things work best when Mike (and the rest of the team) figure out that the daily standup helps them plan the next day’s worth of work—and every single person on the team is part of the planning process.

That makes sense! Planning a project works a lot better when everyone on the team is engaged. But I bet this only works if everyone at the meeting is tuned in the whole time.

How do you change the mindset in a team or an individual? Can you think of examples from your own projects where someone’s mindset changed—maybe your own?
So what is agile, anyway?

Agile is a set of methods and methodologies that are optimized to help with specific problems that software teams run into, and kept simple so they’re relatively straightforward to implement.

These methods and methodologies address all of the areas of traditional software engineering, including project management, software design and architecture, and process improvement. Each of those methods and methodologies consists of practices that are streamlined and optimized to make them as easy as possible to adopt.

Mindset versus methodology

Agile is also a mindset, and that’s a new idea for a lot of people who haven’t worked with agile before. It turns out that each team member’s attitude toward the practices they use can make a huge difference in how effective those practices are. The agile mindset is focused on helping people share information with each other, which makes it much easier for them to make important project decisions (rather than just relying on a boss or project manager to make those decisions). It’s about opening up planning, design, and process improvement to the entire team. To help everyone get into an effective mindset, each agile methodology has its own set of values that team members can use as a guide.

BRAIN POWER

What happens if one team member checks out during the daily standup and doesn’t really listen to his or her teammates?
Here are a few problems that Kate, Ben, and Mike brought up during a daily standup. Across from them, we’ve written down the names of a few different practices that you’ll often see used on agile teams. Don’t worry if you haven’t run across some of them—you’ll learn a lot more about them later in the book, so we included a brief description of each practice to help you out. See if you can match each problem with a practice that might help fix it.

“WE JUST WASTED HOURS GRINDING THROUGH SPAGHETTI CODE TO FIND THAT BUG!”

A retrospective is a meeting in which everyone talks about how the last part of the project went, and talks about what lessons can be learned.

“OK, WE’VE GONE OVER THE USER STORIES. NOW LET’S FIGURE OUT HOW THEY FIT TOGETHER SO WE CAN PLAN OUT THE NEXT FEW WEEKS OF WORK.”

A user story is a way to express one very specific need that a user has, usually written out as a few sentences on a sticky note or index card.

“WE ALWAYS SEEM TO RUN INTO THE SAME KIND OF PROBLEMS OVER AND OVER AGAIN WITH EVERY RELEASE.”

A task board is an agile planning tool in which user stories are attached to a board and categorized into columns based on their status.

“I JUST DEMOED A NEW FEATURE TO ONE OF OUR VIDEO STREAMING USERS. SHE SAID IT WON’T ACTUALLY FIX THE PROBLEM IT’S SUPPOSED TO ADDRESS FOR HER.”

A burndown chart is a line chart, updated daily, that tracks the amount of work left on the project, “burning” down to zero when the work is done.

“I THOUGHT WE’D BE DONE UPDATING THE SONG DATABASE CODE BY FRIDAY. NOW YOU’RE TELLING ME THAT IT’LL BE THREE MORE WEEKS?”

Developers fix code problems by constantly refactoring their code, or improving the code structure without changing its behavior.

It’s OK if you haven’t run across these practices yet. You’ll learn more about them in the next few chapters.
There are many ways that teams can be agile, and there’s a long list of methods and methodologies that agile teams use. But there have been many surveys done over the years that have found that the most common approach to agile is Scrum, a software development framework focused on project management and product development.

When a team uses Scrum, every project follows the same basic pattern. There are three main roles on a Scrum project: the Product Owner (like Ben) works with the team to maintain a Product Backlog; the Scrum Master helps guide the team past roadblocks; and the Development Team members (everyone else on the team). The project is divided into sprints, or cycles of equal length (often two weeks or 30 days) that follow the Scrum pattern. At the start of a sprint, the team does sprint planning to determine which features from the Product Backlog they’ll build during the sprint. This is called the Sprint Backlog, and the team works throughout the sprint to build all of the features in it.

Every day the team holds a short meeting called the Daily Scrum. At the end of the sprint, working software is demonstrated to the product owner and stakeholders in the sprint review, and the team holds a retrospective to figure out lessons they’ve learned.

We’ll cover Scrum in depth in Chapters 3 and 4, and we’ll not only teach you how it helps teams build better software and run more successful projects, but we’ll also use it to explore important concepts and ideas shared by all agile teams.

XP and Lean/Kanban

While Scrum is the most popular agile methodology, many teams take other approaches. The next most popular methodology is XP, a methodology focused on software development and programming that’s often used in combination with Scrum. Other teams approach agile with Lean and Kanban, a mindset that gives you tools to understand the way you build software today and a method that helps you evolve to a better state tomorrow. You’ll learn about XP and Lean/Kanban in Chapters 5 and 6.

Getting a little overwhelmed with new vocabulary?

We’ll highlight new vocabulary words in boldface when we first introduce them. We did that a lot on this page—and if there are a few that you’re not familiar with, that’s OK! Seeing new ideas in context now will help them sink into your brain when you learn about them in detail later. That’s part of the Head First neuroscience that makes this book brain-friendly!
“One of my programmers just wasted hours grinding through spaghetti code to find a bug!”

A task board is a great way for everyone on the team to see the same big-picture view of the project.

“Ok, we’ve gone over the user stories. Now let’s figure out how they fit together so we can plan out the next few weeks of work.”

“A user story is a way to express one very specific need that a user has, usually written out as a few sentences on a sticky note or index card.

“We always seem to run into the same kind of problems over and over again with every release.”

A retrospective is a meeting in which everyone talks about how the last part of the project went, and talks about what lessons can be learned.

“I just demoed a new feature to one of our video streaming users. She said it won’t actually fix the problem it’s supposed to address for her.”

When everyone on the team understands the users and what they need, they do a better job of building software that users love.

“I thought we’d be done updating the song database code by Friday. Now you’re telling me that it’ll be three more weeks?”

A burndown chart is a line chart, updated daily, that tracks the amount of work left on the project, “burning” down to zero when the work is done.

This is an XP practice. Some project managers are surprised the first time they discover agile practices that are focused on code, and not just on planning and executing the project.

Developers fix code problems by constantly refactoring their code, or improving the code structure without changing its behavior.

Teams can avoid making the same mistakes over and over again by looking back at the project and talking about what went right and what could be improved.
**agile is not just new names for old things**

**Dumb Questions**

**Q:** It sounds like Scrum, XP, and Lean/Kanban are very different from each other. How can they all be agile?

**A:** Scrum, XP, and Lean/Kanban focus on very different areas. Scrum is mainly focused on project management: what work is getting done, and making sure that it’s in line with what the users and stakeholders need. XP is focused on software development: building high-quality code that’s well-designed and easy to maintain. Lean/Kanban is a combination of the Lean mindset and the Kanban method, and teams use it to focus on continually improving the way that they build software.

In other words, Scrum, XP, and Lean/Kanban are focused on three different areas of software engineering: project management, design and architecture, and process improvement. So it makes sense that they would have different practices—that’s how they differ.

In the next chapter you’ll learn about what they have in common: **shared values and principles** that help teams adopt an agile mindset.

**Q:** Isn’t this all just stuff I know already, only with a new name? Like, Scrum sprints are really just milestones and project phases, right?

**A:** When you come across an agile methodology like Scrum for the first time, it’s really common to look for the parts of it that are similar to things you already know—and that’s a good thing! If you’ve been working on teams for a while, then a lot of agile *should* feel familiar. Your team builds something, and you and your teammates are almost certainly doing a lot of things well that you don’t want to change (yet!). However, it’s very easy to fall into the trap of thinking that a familiar-seeming part of agile is exactly the same thing as something you already know. For example, Scrum sprints are *not* the same thing as project phases. There are many differences between phases or milestones in traditional project management and sprints in Scrum.

For example, in a typical project plan, all of the project phases are planned at the beginning of the project; in Scrum, only the next sprint is planned in detail. This difference can feel very strange to a team accustomed to traditional project management.

You’ll learn a lot more about how Scrum planning works, and how it may be different from what you’re used to. In the meantime, keep an open mind—and try to catch yourself when you have the thought, “This is just like something I already know!”

**BULLET POINTS**

- Many teams that want to adopt agile start with the **daily standup**, a meeting with the whole team where everyone stands in order to keep it short.

- Agile is a set of **methods and methodologies**, but it’s also a **mindset**, or an attitude that’s shared by everyone on the team.

- The daily standup meeting is much more effective when everyone on the team has the right **mindset**—everyone listens to each other, and they all work together to make sure the project is on track.

- Every agile methodology comes with a set of **values** to help the team get into the most effective mindset.

- When team members follow shared **principles** and share the same set of **values**, it can make the method that they use **much more effective**.

- **Scrum**, a framework focused on project management and product development, is the most common approach to agile.

- In a Scrum project, the team breaks the work into **sprints**, or cycles of equal length (often 30 days) that follow the Scrum pattern.

- Every sprint starts with a **sprint planning** session to determine what they’ll build.

- During the sprint the team works on the project, and every day they hold a short meeting called a **daily scrum**.

- At the end of the sprint, the team holds a **sprint review** with the stakeholders to demo the working software that they built.

- To finish the sprint, the team holds a **retrospective** to look back at how the sprint went and discuss ways that they can improve together as a team.
Don’t just dismiss the idea that mindset matters!

A lot of people—especially hardcore developers—tune out as soon as they start hearing words like mindset, values, and principles. That’s especially true of coders who have a habit of locking themselves in a room and never talking to anyone. If you’re starting to think this way, really try to give these ideas the benefit of the doubt. After all, a lot of great software has been built this way, so there has to be something to it... right?

Sharpen your pencil

Which of these scenarios are examples of applying practices, and which are examples of applying principles? Don’t worry if you haven’t seen some of these practices yet, just use the context to try to figure out the right answer. (That’s a good skill to work on for taking a certification exam!)

1. Kate knows that the most effective way to communicate important information about the project to her team is with a face-to-face conversation.
   - Principle
   - Practice

2. Mike and his team know that the users will probably change their minds later, and those changes can wreak havoc with their code, so they use incremental design to make sure the code that they build is easy to change later.
   - Principle
   - Practice

3. Ben uses a persona to model a typical user because he knows that the more the team understands the users, the better job they’ll do of building software.
   - Principle
   - Practice

4. Mike always makes sure that his team is working on something that he can demonstrate to Kate and Ben, because he knows that working software is the best way to show the team’s progress.
   - Principle
   - Practice

5. Kate wants to improve the way that the team builds software, so she gets them all to improve collaboratively and evolve experimentally by coming up with changes that they can make to their process together, and using data to figure out if those changes made things better.
   - Principle
   - Practice

6. Mike and his team embrace change by building code that’s easy to change down the road.
   - Principle
   - Practice

Answers on page 20
more visibility is better (right?)

Wow!
WE'VE NEVER WORKED TOGETHER THIS WELL BEFORE. THAT DAILY STANDUP MEETING CHANGED EVERYTHING!

Kate: This project’s gone so much better than ones in the past. And all because of one little meeting every day!

Mike: Well, I wouldn’t say that.

Kate: Come on, Mike! Don’t be such a pessimist.

Mike: No, seriously. Look, you didn’t really think you’re the first person to try to solve our project problems by adding meetings, did you?

Kate: Well, I... um...

Mike: We got some really good results, so I’ll be honest with you here. When you started holding those daily standups, almost everyone on the team was unhappy.

Kate: Really?

Mike: Yeah. Don’t you remember how for the first week and a half, most of us just stared at our phones the whole time?

Kate: Well, sure. I guess that wasn’t particularly useful. If I’m honest with myself, I was actually thinking about calling the whole thing off.

Mike: But then one of my coders brought up that serious architecture issue. Everyone listened because she’s really good, and they all respect her opinion.

Kate: Right. We had to make a major change, and I cut two of the features out of the release to make room for it.

Mike: Yes! That was really important. Normally when we run into a problem like that, we have to work late nights to deal with the aftermath. Like when we found out that the listener feedback analysis algorithm had a serious flaw in it.

Kate: Ugh, that was awful. I usually find out about problems like that after we’ve all promised things we couldn’t deliver. This time we caught the problem early, and I could work with Ben to manage our users’ expectations and get you guys the time you needed to come up with a new approach.

Mike: We’ll definitely bring up problems like that every time they come up.

Kate: Wait—what? That kind of problem happens a lot?!

Mike: Are you kidding? I’ve never had a project that didn’t run into at least one nasty surprise like that once we started coding. That’s how software projects work in the real world, Kate.