RICHARD LAWRENCE
with PAUL RAYNER

Behavior-Driven Development *with* Cucumber

Better Collaboration for Better Software

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Behavior-Driven Development with Cucumber

This page intentionally left blank

Behavior-Driven Development with Cucumber

Better Collaboration for Better Software

Richard Lawrence
with Paul Rayner

Addison-Wesley

Boston • Columbus • New York • San Francisco • Amsterdam • Cape Town • Dubai
London • Madrid • Milan • Munich • Paris • Montreal • Toronto • Delhi • Mexico City
São Paulo • Sydney • Hong Kong • Seoul • Singapore • Taipei • Tokyo

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2019937018

Copyright © 2019 Pearson Education, Inc.

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit www.pearsoned.com/permissions/.

ISBN-13: 978-0-321-77263-3
ISBN-10: 0-321-77263-6

1 19

Cover photograph © Food Travel Stockforlife/Shutterstock

Publisher
Mark L. Taub

Acquisitions Editor
Haze Humbert

Development Editor
Ellie Bru

Managing Editor
Sandra Schroeder

Senior Project Editor
Tonya Simpson

Indexer
Erika Millen

Proofreader
Abigail Manheim

Technical Reviewers
Leslie Brooks
Matt Heuser
Nicole Forsythe
Matt Wynne

Cover Designer
Chuti Prasertsith

Compositor
codeMantra

Contents

Acknowledgments

Writing this book was far and away the most difficult professional challenge I've ever undertaken. The book would not exist without the help and support of many people in my life.

I'm grateful to my business partner at Agile For All, Bob Hartman, for believing I had something to say on this topic that was worth a book and for getting the ball rolling on it.

Bob introduced me to Chris Guzikowski at Pearson, who took a risk on a new author and showed remarkable patience when client work often got in the way of writing.

I'm thankful for my friend Paul Rayner, who stepped in for a time as co-author when the book was stalled and helped transform it to be much more interesting and useful. Even after leaving the project to focus on other things, Paul still read drafts and offered invaluable feedback.

My wife, Dawn, provided endless patience and support throughout this long project. She read drafts, talked through ideas and challenges with me, and most of all sustained me with unwavering belief that I actually could see the project through. Thank you, Dawn.

My dad, Tom Lawrence, introduced me to software development as a child. He encouraged me and helped me grow early in my career. And then he was willing to take my ideas, apply them in his work, and help me make them better. To be able to become a peer and even a coach to the man who inspired me to do this work at all is a remarkable thing, and I'm so grateful for it.

I'm thankful to the many people who were willing to read unpolished drafts and sacrifice time to contribute detailed feedback. Thank you, Ingram Monk, Kim Barnes, Colin Deady, Dan Sharp, Sean Miller, Jen Dineen, David Madouros, Josh Adams, David Watson, Seb Rose, Aslak Hellesøy, Brennan Lawrence, Donna Hungarter, Nicole Forsythe, Matt Heusser, Matt Wynne, and Leslie Brooks. (And apologies to anyone I missed in that list—so many people read bits of the book over the eight years we worked on it.)

This is a book rather than just a bunch of words sitting on my computer because of the many great people at my publisher, Pearson. Thank you to Chris Zahn, Payal Sharotri, Mark Taub, Haze Humbert, and all those behind the scenes who I'll never interact with directly. Special thanks to my development editor Ellie Bru and

copyeditor Tonya Simpson for carefully reading the book multiple times and helping me get the thoughts clear and the words just right.

My Agile For All colleagues have been encouraging, patient, and supportive. I'm thankful to work with such an amazing group of people.

I'm indebted to many in the BDD and Agile Testing communities who have shaped my thinking on these topics, including Ward Cunningham, Kent Beck, Ron Jeffries, Dan North, Liz Keogh, Matt Wynne, Aslak Hellesøy, Seb Rose, Gaspar Nagy, Lisa Crispin, Dale Emery, Elisabeth Hendrickson, Gojko Adzic, and Jerry Weinberg.

Finally, a huge thank you to my clients who have tried, challenged, and often extended my ideas. I learn from you every time we work together.

—Richard Lawrence

About the Authors

Richard Lawrence is co-owner of the consulting firm Agile For All. He trains and coaches people to collaborate more effectively with other people to solve complex, meaningful problems. He draws on a diverse background in software development, engineering, anthropology, and political science.

Richard was an early adopter of behavior-driven development and led the development of the first .NET version of Cucumber, Cuke4Nuke. He is a popular speaker at conferences on BDD and Agile software development.

Paul Rayner co-founded and co-leads DDD Denver. He regularly speaks at local user groups and at regional and international conferences. If you are looking for an expert hands-on team coach and design mentor in domain-driven design (DDD), BDD with Cucumber, or lean/agile processes, Paul is available for consulting and training through his company, Virtual Genius LLC.

This page intentionally left blank

Chapter 2

Exploring with Examples

As we said in Chapter 1, "Focusing on Value," the first and most important part of BDD is exploring desired system behavior with examples in conversations.

We set the stage for conversations around examples by finding valuable slices of functionality on which to focus in the form of MMFs and user stories. In this chapter, we look at how to explore the behavior of a new user story by collaborating around examples with a cross-functional team.

Involving the Team to Describe a Story with Examples

First Whole Team Meeting

ROBIN: A developer. Joined the team a month ago after working for several startups in the Bay Area. She's familiar with Agile and has a voracious appetite for learning the latest tools, techniques, and applications.

RAJ: Senior developer on the team. Has an MSc in Computer Science and has been working in library systems for many years.

JANE: Tester. She has a poster in her cube that says, "A failure to plan on your part does not constitute an emergency on mine." Very focused on her work and enthusiastic about trying things that can help her be more effective and efficient.

JESSIE: ScrumMaster. Started out as a web designer, but after seeing another ScrumMaster in action she realized she might enjoy trying that. Fantastic at digging in and helping the team solve their impediments, often so seamlessly that the team doesn't realize she did it, attributing it to coincidence or themselves. She's OK with that.

SAM: Business analyst. Sam is a late adopter, very pragmatic, structured, and process-oriented. He was attracted to becoming a BA because he likes putting things in order. After a few beers he might tell you he understands the library processes better than the librarians do.

Setting: A small conference room in the library's downtown office. There is a conference table with eight chairs and a pile of sticky notes, and an empty whiteboard along one wall.

(MARK, ROBIN, RAJ, JANE, JONAH, JESSIE, and SAM have assembled in the conference room and are waiting for MARK to start the meeting.)

MARK (Product Owner): Hi, everyone. As you all know, we've developed a lot of great software and generally have done a good job of keeping most of our library patrons happy. In our last project retrospective, we said we wanted to get better at communicating about scope and reducing rework. This new project seemed like a good opportunity to experiment with a new way of working together.

I mentioned yesterday that I would be bringing in Jonah to help coach us in some new approaches and techniques to help us collaborate and communicate better. Since you've all just come off a successful release, Susan, Jessie, and I agreed it might be best to treat what we do here as an experiment, as a "proof of concept" for these techniques.

We'll still be developing production-ready software but "kicking the tires" on learning BDD with Cucumber as we do it. So, we've set aside the next week or so to work on our first feature, with coaching help from Jonah along the way. It will be outside our regular sprint approach to give us a little more freedom to experiment and learn. How does that sound to everyone?

(Nods of agreement around the room)

RAJ (Developer): Sounds great. This approach gives us a chance to try some things without the regular pressure of deadlines and make mistakes as we learn. Plus, we're still delivering real features.

I've been doing Scrum for so long now, though, it will feel a little weird to not follow the sprint structure.

JESSIE (ScrumMaster): Agreed. It's just something Mark and I thought might help you all feel more comfortable as you ramp up. Let's still meet for our regular daily standups, but we'll treat demos as something we do once we feel like we're ready, rather than trying to work to a regular sprint schedule.

MARK (Product Owner): Works for me. Looks like we're all in agreement.

Jonah, you had a conversation with Robin, Raj, and Jane yesterday afternoon about tooling for this, right? I assume we're all on board with using Cucumber to support testing the features as we develop them.

JONAH (Coach): Thanks, Mark. Yes, yesterday I stopped by the team area and Robin, Raj, Jane, and I talked through the change in approach and new tooling that you'll be experimenting with as a team. They've also each watched a BDD overview video online I had pointed out to them.

ROBIN (Developer): I can't wait to try this out! Cucumber looks very cool and I've wanted to play with Capybara for a long time now but not had the chance.

RAJ (Developer): It looks intriguing.

JANE (Tester): I'm really hoping this will mean I won't have to deal with lots of functionality to test at the end of each sprint like the last release, but we'll see. It should at least reduce the amount of manual testing I do and help with regression testing in the future.

MARK (Product Owner): Sam, how about you?

SAM (Business Analyst): I don't know what you're all talking about.

MARK (Product Owner): What?

JONAH (Coach): Sam, I apologize. I tried to loop you into yesterday's conversation, but your office door was shut and it looked like you were meeting with someone at the time. It's unfortunate that you weren't able to be part of the introduction. Are you available after this meeting to talk through any questions you might have?

SAM (Business Analyst): Sure, no problem.

JONAH (Coach): Excellent. I'll also talk you through the same things I went through with the others and send you a link to the same video.

What we're going to practice in this meeting is having a conversation together about a new feature for the library website, talking about what "done" means for that story, and gathering some examples that we'll later turn into test scenarios.

MARK (Product Owner): Help me understand, Jonah. Is what you just described the essentials of the "BDD process" you described to me earlier?

JONAH (Coach): Conversations about the user goals for features, examples of how the business works and how that feature fits into the business processes are an important part of BDD for sure. We'll be using business examples to help us have a shared understanding of software we need to build. There's more to BDD than just those things, though.

One comment I'll make is about BDD being a "process." What comes to mind when you think of "process?"

RAJ (Developer): I think of something heavyweight, like RUP. Something that has a lot of steps and roles and approvals.

MARK (Product Owner): Before we adopted Scrum, our process was really heavyweight: many approvals and lots of waste, and lots of competition over scope between the business and our teams. We tried to keep the scope under control, and the business tried to cram everything they could into the requirements document because they got only one shot at it. But, in terms of Agile, when I think of process I think of Scrum. It has a minimal set of roles, artifacts, and meetings. It can seem like a lot, but it's not really. So process doesn't have to be a bad word.

ROBIN (Developer): Right, me too. I think of overhead, and having to do things just because the process says so, even when it doesn't make sense. One thing I liked about doing Agile in the startups I used to work for is there was minimal process overhead compared with the first job I had out of college.

JONAH (Coach): What about a game? What comes to mind when I say the word "game?"

JANE (Tester): Something like basketball, I suppose. A team of people working together, playing positions, helping each other, trying to win against the other team.

SAM (Business Analyst): A game has a goal, like winning against the other team. And you need skilled players used to playing together to achieve the goal. The players have to adjust their strategy on the fly as the game progresses.

ROBIN (Developer): A game should be fun.

JONAH (Coach): You've got it. Why do I ask this? Mainly because I view BDD much more like a game than a process. BDD has a goal, it takes a team working together, team members need to grow their skills over time with individual and team practice, and the team positions and strategy are fluid as the game progresses. And BDD, like any game worth playing, can be very challenging and demanding, but the rewards are worth it. As the team improves over time, playing the BDD game should be fun and fulfilling.

SAM (Business Analyst): But games are trivial things; they are for children. We should be software professionals, not "software hedonists."

JONAH (Coach): Right. The goal isn't the fun, that's a good side effect, though I do believe people who enjoy their work are much more likely to be productive than those who don't. Don't mistake the word "game" for something trivial or just for children, though. Games are invented and used by many people, including novelists, military tacticians, mathematicians, game theorists, and corporate strategists.

So, think of BDD more like a game than a process as we move forward. Like any game, it takes a bit of practice to learn it. And you shouldn't expect to be good at it right away, especially since it involves discerning goals and learning to work more closely together as a team to achieve those goals.

Let me say again, in this meeting we'll be trying an approach that is going to feel new and different, maybe even weird and counterintuitive, to most of you at first, so I'd encourage you to roll with it and see where it takes us. Have fun with it. If any of you have concerns or questions then feel free to grab me later and I'll do my best to address them.

BDD Is a Cooperative Game

In Alistair Cockburn's book *Agile Software Development: The Cooperative Game*, he characterizes software development as a "cooperative game of invention and communication." In competitive games, like tennis, there is a clear notion of winning and losing. Even in team games, like basketball, one team wins and the other loses. But in cooperative games, people work to win together.

Games can also be finite or infinite. A finite game, like chess, is one that intends to have an end. On the other hand, an infinite game, like the game an organization or nation typically plays, is about prolonging one's existence.

Games can be finite and goal-seeking, like chess, or they can be finite and non-goal-directed, like jazz, where the process is the focus—there's not a defined goal that would cause you to "win" the song.

If we combine these ideas to get a goal-seeking, finite, cooperative game, we see activities like rock climbing or adventure racing, where a group of people work together to reach a goal together as fast as possible. Software development, especially Agile software development, is a similar game. *Software development is a finite, goal-seeking, cooperative, group game.*

BDD is an Agile subdiscipline of the game of software development. The emphasis in BDD is particularly placed on helping the team cooperate, innovate, and communicate during the game, all with the intention of achieving the goal of creating valuable working software more quickly and effectively.

BDD Is a Whole Team Thing

Teams who find their way into BDD via a tool like Cucumber often get the misunderstanding that BDD is a test automation approach that mostly concerns testers and developers.

BDD is a whole-team practice. It's a way of structuring the collaboration required to build the right software. As such, it involves all the roles on the team.

Product owners bring an understanding of the customer and realistic examples of a user doing something with the software. Testers bring their unique perspectives about what could go wrong; they're great at proposing examples outside the happy path. Developers understand the implications of a particular example on implementation and often have a good understanding of the existing system and the problems it solves. Technical writers contribute skills with language and often empathy for how users talk about what they do in the system.

Early in adopting BDD, collaboration tends to occur as a whole team. This allows the team to build a common language for their domain. Later, smaller groups can leverage that language to collaborate, and the work they produce will be comprehensible to the rest of the team.

Because we see so much value in the whole team participating in the collaboration in BDD, we rarely teach public BDD classes. If only some team members understand how to work in this way, the team is unlikely to experience the benefits. We've seen cases where just the testers or just the developers get excited about a tool like Cucumber and just end up doing test automation, the least valuable part of BDD, in isolation.

Allow Time and Space to Learn

Did you notice how Jonah warned the team it wasn't going to be easy? Adopting a new way of working takes time and practice. It takes a willingness to muddle through until the new skills become natural. It's easy to forget what we went through to adopt our current skills—they weren't always as natural as they seem now.

Sometimes it feels like you can't afford the time to learn something new. Maybe someone committed you to deliver a big project (with defined scope) by a particular date, and it doesn't feel like you have any room to slow down. In that case, feature mining from Chapter 1 is your secret weapon.

These big release or project commitments that feel so fixed and overwhelming are usually fairly high-level. We know we have to deliver, say, the new sales reporting by June 15. But within the bullet point of "new sales reporting," there's high-value work and there's low-value work. Feature mining gives us a way to focus on the high-value work and avoid the low-value work, thereby making the deadline less scary and buying some extra capacity for improvement.

We also don't recommend adopting a new practice like BDD for all your work all at once. Instead, we recommend you use what we call the "slow lane" approach. Say you've planned eight user stories in your sprint. Most stories will use your existing definition of done, which doesn't include BDD. Choose one or two to be in the "slow lane" across your board (see Figure 2-1). For those, you'll use the new practice. And you won't feel as pressured when you do it because you've already agreed those will take longer.

The early adopters on your team who are excited about the new practice will tend to sign up for those stories, working out the issues and establishing patterns for those who prefer to "wait and see."

After a while, expand the slow lane to three or four stories. Eventually, you'll notice that the stories with the new practice really aren't that much slower (and probably turn out faster in the end because they have fewer defects and less rework). At that point, make the new practice part of your definition of done for every story.

Some teams move from the slow lane to fully adopting BDD in just two or three sprints. Others take their time and do it over four to six months. There's no rush. It's better to adopt a practice slowly, deliberately, and well than to rush into it badly and give up under pressure.

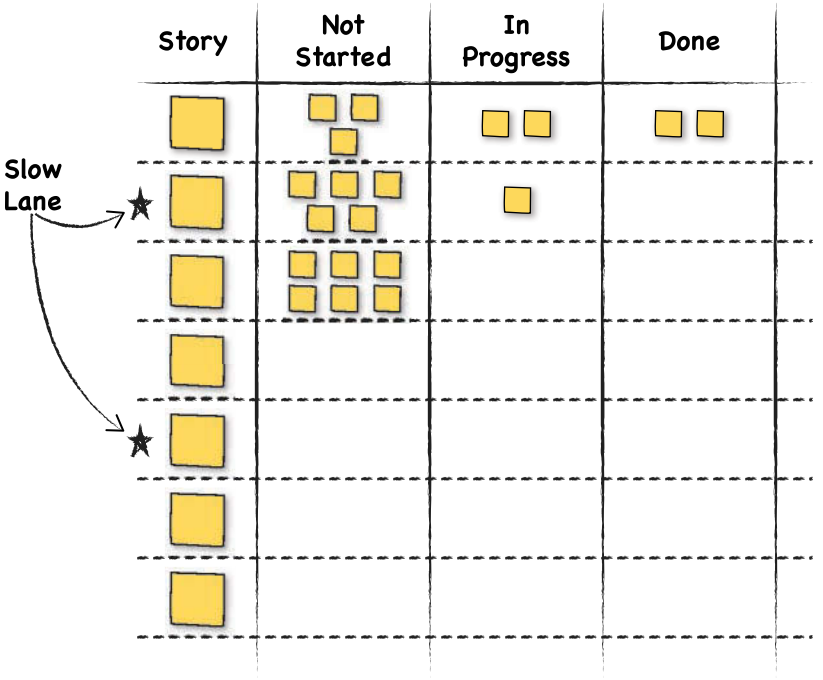

Figure 2-1 *A task board with two stories designated as slow lane stories*

The Meeting Continues…

JONAH: Mark, how about you start by introducing the feature you'll be developing first?

MARK: Sure. Thanks, Jonah.

Well, the first minimal marketable feature, or MMF, is about borrowing an ebook from Amazon. Our first story is implementing searching for an ebook.

JONAH: Let's use sticky notes to put some structure around our discussion. Mark, use one of these yellow rectangle stickies to capture the story at the top of the whiteboard.

MARK: *(Writes on sticky note)* In order to find the specific ebook I'm looking for, as a library patron, I want to search the ebook catalog by title. *(Sticks it to the whiteboard)*

ROBIN: We've been using user stories in our backlog, but they're a little different in form from this one. I like how you put the "In order…" first to highlight the value to the library patron as a customer-facing feature. This is nice.

JONAH: Good observation, Robin. That format is all about focusing on "why" rather than just "what." I'd encourage you all not to focus too much on the format, though. A user story should function as a reminder to have a conversation about the user need being met.

Mark, why don't you give a little background on why we are going to focus on this story in particular?

MARK: Sure. We've seen that people search on title about 80% of the time. So, they know which book they want to find. Currently, the catalog search returns too many unrelated titles in its results, and even when the correct title is returned near the top of the results, it takes too long, and the results are presented in a way that's confusing to our patrons. It's no surprise we've had complaints from library patrons about not being able to find books we actually have in the catalog.

What we want to do is dip our toes in the waters of digital materials but do it with a minimum investment in custom software development. We want to offer a small set of ebooks and only for Kindle. That way, Amazon will take care of the delivery and expiration of the books and we don't need to worry about that. So, Susan and I agreed we could start by building out a new search by title for Kindle ebooks to see how our patrons use it.

JONAH: I wanted to let you know that Raj, Mark, and I talked yesterday and instead of rewriting your legacy system, you'll be adding these new capabilities over the top of the existing system. Taking this approach will dramatically reduce your risk while enabling you to get the new work done quickly, since you won't be bogged down in trying to improve code that's not associated with new value you're adding.

RAJ: Yes, after talking with Mark earlier about this MMF I had expected to spend a couple of weeks sketching out the architecture for a complete ebook management system or figuring out how to rewrite the current catalog module in our legacy integrated library system.

I've wanted to rewrite our legacy ILS for years, but I always suspected we'd never finish it and would soon find ourselves with *two* legacy ILSs to support.

JONAH: *(Laughs)* Yes, that's the typical outcome of that approach. Avoid it wherever possible. Better to take a more strategic approach to replacing capabilities within an older system.

RAJ: Instead of rewriting, we're going pull the Kindle ebook metadata directly from Amazon and use it to enrich our existing catalog in our ILS, rather than try to track the digital inventory separately from our current ILS catalog inventory. So, the first thing is to augment our legacy system rather than try to rewrite it. That will save us a lot of time and minimize our risk, especially when we are trying to learn BDD as well.

JONAH: A second approach you're taking that will help a lot is to leverage open source for the generic parts of your system—right, Raj?

RAJ: Yes, absolutely. I've found a great open-source library to use for search. The catalog team has already configured the backend system to exclude ebooks from regular searches. Jonah and I sketched it all out on the whiteboard in the team area if anyone's interested.

JONAH: Raj was able to come up with a really elegant and clean way of integrating this open-source search framework with your current ILS, avoiding having to write your own search code or try to rewrite any of your ILS. Double win!

ROBIN: That's great news.

JONAH: Let's capture assumptions and questions on pink stickies next to the story to keep us focused. I've heard three key assumptions so far. *(Writes on stickies as he talks)*

Using Amazon Kindle ebook metadata

Adding to the current ILS

Using OS search framework

(Sticks them to whiteboard next to the story card)

ROBIN: OK, dumb question time. I'm looking at the story and, as you know, I'm pretty new on the team and haven't done development work for a library before, so I want to be clear on what you mean by "title."

Are you talking about the *name* of a book or what comes back in the search results, or is it something else? And how does this work with ebooks? Are you meaning Kindle, epub, mobi, audiobooks, mp3…or are there other formats?

MARK: No, those are fundamental library concepts we all need to understand. There's no dumb question there.

JONAH: Robin is asking some great questions about the library domain. I suggest that rather than ask for definitions, let's focus on specific examples. For example, let's pretend Robin asked, "I don't really know what you mean by *title*—can you give me a typical *example* of an ebook title?"

SAM: Well, there are actually some nuances about *title* that we don't really need to worry about now. Mark, I suggest for now we just use *title* to be the name of the book, and we can come back later and talk about the other ways *title* gets used around here.

ROBIN: No problem.

JANE: Actually, Jonah's suggestion about using examples is a good one. It would help us all get on the same page about how the search needs to work. I'm typically working off specific examples in my current test plans.

JONAH: Right. To build on that a little: Since we are talking about Kindle ebooks… Mark, for starters, can you give us an example of an ebook that one of our library patrons would actually be searching for?

MARK: OK. Sure, hmmm. What's a good example? Let's say fantasy fiction author Brandon Sanderson releases a new title in his *Stormlight Archive* series and it gets released on Kindle. Each of these has been a bestseller in the past, so we know any new book in the series is going to be in very high demand. For example, *Words of Radiance* debuted at #1 on the *New York Times* Hardcover Fiction Bestseller list in early 2014, and the ebook reached #1 on the combined print/ebook bestseller list. So, how about we use *Words of Radiance* as our example?

ROBIN: Good choice! I've been enjoying that series.

(JANE picks up a marker to write the example. JONAH hands Jane a pad of green sticky notes.)

JANE: *(Writes "Find an ebook by the exact title: Words of Radiance" and sticks it on the whiteboard.)*

I'm not much of a fantasy reader, but *(Looking meaningfully in Robin's direction)* I have friends who are. Sounds like a good one to start with. Let me capture this on the whiteboard.

(Writing on a pink sticky note) I'll also capture Robin's earlier question about titles, so we can get that up on the whiteboard for later too.

SAM: So, we're assuming we already have *Words of Radiance* in the catalog, right? Cause if we don't, then the search results would be different.

MARK: Well, we need to handle the situation where it doesn't show up in the search results. That's part of assumptions for this story.

JONAH: Sam and Mark both have a good point, I'd encourage you to stick with the simple case for now—the "happy path"—so we can dig into its nuances. We can capture any other scenarios and come back to them later.

Let's capture the rule for the happy path. I'll call it "Matches title word for word" and put that above the example on the board. *(Writes on a blue sticky note)*

This makes me want the counter example. I'll call it "Doesn't find the book: *Words Radiance*". *(Writes on a green sticky note)*

JANE: I just noticed something: Since *Words of Radiance* will be in the catalog, this choice of example means we won't have to do test data setup and teardown for this story. We can simply test against the real catalog because the Amazon Kindle ebook metadata will have been loaded into it.

JONAH: It's a nice outcome from taking Raj's approach. It's not a long-term approach, of course—we're going to need to figure out how to get the catalog into a particular state for scenarios later—but it will help you get up and running quickly without getting bogged down in technical issues.

At this point, you have complexity in a lot of areas: you're building a whole new set of capabilities for your customers, solving new technical problems, and exploring a new way of working together. Deferring technical complexity for now is a nice way to get to customer value faster. However, we'll want to set up a time soon to talk about how you'll handle the data setup problem when you get there.

RAJ: Yes, it's a big relief. I was worried about how we would handle the whole test data side of things.

JONAH: Exactly. Let's not bite off too much infrastructure stuff at this point. We'll deal with it soon enough.

JESSIE: OK, I'll capture that "missing title" scenario for later reference then. *(Writes "Search for an ebook by exact title but missing from our catalog" on a sticky note and puts it on the whiteboard)*

Flesh Out the Happy Path First

It's often easy to come up with a bunch of examples right away, but we recommend talking all the way through at least one core example before getting deep in variations. By talking through a "happy path" example, a common case where things work as they should, the team gets a shared understanding of what the user story is about. What does it look like when the user is able to do what they want to do?

This complete slice also helps validate the story. Sometimes a story that sounds quite reasonable in the abstract reveals mistaken assumptions when you get into a core example. "Wait a minute," someone says, "no user would actually do it that way."

Once everyone understands the happy path, it often becomes more clear which variations are reasonable and likely. Instead of brainstorming every possible variation and edge case, realistic examples float to the top and focus the team on getting to value quickly. (The less likely variations and edge cases can be useful for exploratory testing, by the way. More on this and how it relates to BDD in Chapter 3, "Formalizing Examples into Scenarios.")

Use Real Examples

Notice the team's happy path example wasn't just about "a book" or "Title ASDF"; it was an example of a real book that a real library patron might borrow. Good examples put you in the user's shoes, building empathy for what the user is trying to do and helping you think more accurately about how they'll do it.

It wasn't a coincidence that using *Words of Radiance* for the example led the team to realize they already had test data to use. A real example connected the team to the system full of real inventory they already had at the library. This is not unusual. One real thing connects to other real things. We frequently see teams make discoveries about functionality, design, and architecture when they get into concrete, realistic examples.

Example Mapping Gives the Discussion Structure

Our favorite way to add some light structure to this kind of discussion is with Matt Wynne's Example Mapping technique.[1] In Example Mapping, the group builds a tree: the story has rules that are illustrated by examples. Assumptions and questions are captured on the side so the core discussion about examples doesn't get blocked by side conversations. The result looks something like Figure 2-2.

Discussions about examples necessarily lead to rabbit trails—side topics that could distract from the core topic—because reality is complex and interlinked. In the meeting we just looked at, the team allowed side topics to naturally emerge but retained focus on the core topic with a facilitation tool called a parking lot. The parking lot is simply a place to capture things that are worth talking about but not right now.

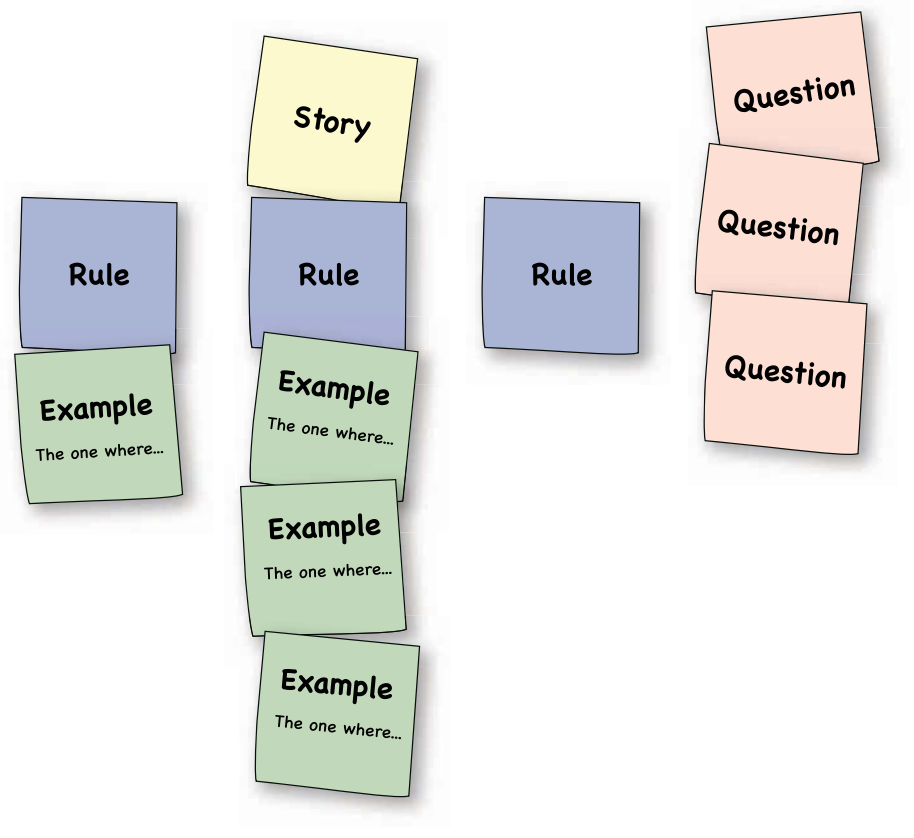

Figure 2-2 *Example Mapping*

1. https://docs.cucumber.io/bdd/example-mapping/

In Example Mapping, pink stickies typically function as the parking lot—the questions and assumptions that otherwise might take over the discussion. But yellow, green, or blue stickies could capture stories, examples, or rules to be included in a future discussion.

Jane captured a note about audiobooks, for example. Audiobooks were explicitly out of scope for the current feature, but they were worth thinking about. Mark will probably take that note, convert it into a product backlog item, and prioritize it appropriately.

Using a parking lot balances the needs to be open to diverse contributions and to capture important discussion topics with the need to respect meeting participants' time by keeping the meeting focused. We use variations on this technique in almost everything we facilitate.

Back to the Meeting…

RAJ: Back to Robin's question again, there are multiple formats we support, so we might have *Words of Radiance* in mobi for the Kindle and epub for Apple, and as an audiobook in mp3.

ROBIN: That's what I thought: The first two are alternatives to getting the hardcover or paperback book, for people looking for electronic reading formats, whereas mp3 is for people who would rather listen to the book.

MARK: You're right that there are multiple formats, but that's not a concern for us with this story. As I mentioned before, Susan and I agreed to focus just on Kindle ebooks for now, so let's keep the reading aspect primary and only worry about Kindle format for now.

(JANE writes "no audiobooks for now" on a pink sticky and sticks it to the whiteboard.)

RAJ: I've done a lot with search in the past, and there are many ways to search our catalog: keyword, author, title, and so on. Are we talking about a keyword search here?

MARK: I actually talked about this earlier with Susan, and when we looked at the traditional searches over 80% of them were by title, and the patron picked the book with the matching title.

JESSIE: *(Writing on a sticky note)* Got it, search by title only.

OK, some user experience kinds of questions here: We are assuming the library patron is on our catalog page, correct?

MARK: Yes, catalog page.

JESSIE: OK, good. Also, I'd like us to talk a little about what we can assume or expect about the patron's typical background, goals, and pain points.

JONAH: That's where I was hoping we would head too. What Jessie is concerned about is moving beyond an impersonal notion of the "role" of the system user to something a bit more human, a "persona."

SAM: What do you mean, persona? Isn't this just a story that says something like, "As a library patron, I want to search the library catalog for an ebook"?

JESSIE: True. Jonah and I talked yesterday about starting to incorporate personas into how we do our user experience work. Basically, a persona is a short description of the important characteristics of a certain kind of user—their goals, background, experience, and pain points. Putting together personas will really help us do a better job of user experience going forward, rather than our UI treating all of our users as if they are the same, which they're not. Using personas encourages us to think more from a customer perspective and avoid implementing features our customers won't really need.

JONAH: Right, as a friend of mine says, "A persona's biggest benefit is not to tell us what to include, but to tell us what *not* to."

Yes, Sam, the story will probably look something very similar to that in the product backlog. A story is a placeholder for a team conversation, so I suggest we take a moment to dig a little deeper and put more of a face to whom we mean by *library patron* for this feature.

Mark, can you give us a little background on the library patron you would like this MMF to target?

MARK: Sure. Most of our search users are familiar with Google search and using websites but not much beyond that. They are used to being able to easily find the book they want on Amazon. We would much rather they come to our library than go somewhere else, and so would they.

You mentioned their goal, and I guess it's to find the book they want with as little hassle, and encountering as little confusing technical jargon, as possible. And we decided in that previous meeting that we

would only target Kindle users for this story, as Kindle is the most popular platform.

JESSIE: *(Sketching notes about persona on whiteboard)* So we can assume some basic technical background for our typical library patron, but not much. And we want to avoid confusing technical or library-specific jargon in the UI where possible. Are most of them coming through a web browser?

MARK: Yes, correct on all counts. Overly technical jargon in search is a current pain point for many of our users.

ROBIN: So when the user, I mean *library patron*, sees the search results, there will be other books listed, but *Words of Radiance* should be at the top? *Words of Radiance* won't be the only result, right?

MARK and SAM: Right.

JONAH: OK, good. So we can make it simple because this first search scenario should talk about having *Words of Radiance* at the top of the search results, but it doesn't need to say anything about what the rest of the search results might be.

MARK: Makes sense to me.

JONAH: It seems like we probably have a good enough shared understanding of the happy path scenarios to stop here. We don't need everyone involved to start formalizing them up a bit for the Cucumber feature file; a few of us can tackle that later.

It would be helpful to take a photo of the whiteboard to capture what we've done so far and send it around. Raj and Robin, would you like me to help you take an initial pass at writing up the scenarios as a feature file?

RAJ: Sure. Jonah, how about you, Robin, and I try doing that right after lunch and we all meet back here to review what we come up with? What do you all think? 3 p.m. work for everyone? Good.

I'll take care of taking the photo of the whiteboard and sending it around too.

MARK: Sounds great. I'll send everyone another meeting invite for this room at 3 p.m. This approach is pretty new to me, so I'm looking forward to seeing what you come up with.

Optimizing for Discovery

You might be surprised that the team proceeds with work on this user story before nailing down all the rules and examples. Rather than working in a series of phases—planning, analysis, development, testing—the team is working iteratively in very small cycles. They're planning and analyzing a little, doing some test design, developing some production code, doing a little more testing, going back to planning, and so on.

The traditional approach to software development assumes we not only have a clear understanding of the goal of a software system but also the means to achieve that goal, and that we know both well ahead of when the software needs to be delivered to the customer. This implies that there is no need to further explore the customer's space, since we already know what the customer requires. The waterfall development process is the logical outcome of this kind of thinking: the means of accomplishing the goal are called *requirements*, and all that is needed is for us to implement the requirements as software features. In industrial work, we want to manage for consistent, repeatable, predictable results; we want our goals to be as clear and unambiguous as possible. Waterfall development takes the same approach as industrial work.

In knowledge work we need to manage for creativity. We are generating something *new* rather than just incrementally improving on the past. This means there is no way to precisely define the goal in advance, because there are too many unknowns. Software development is knowledge work, and Agile development is fundamentally a considered response to this situation of not knowing the exact goal and how to reach it. Agile promotes embracing and adapting to change to be the chief concern of the development process. Thus, teams doing Agile development *optimize for discovery*, as the Manifesto for Agile Software Development says, valuing "responding to change over following a plan." Furthermore, "Agile processes harness change for the customer's competitive advantage."

This ability to respond quickly to new learning is thus at the heart of agility in software development. "The path to the goal is not clear, and the goal may in fact change."[2] An Agile team will often start out with the intention of solving a customer's problem in a certain way but discover their assumptions about that customer need are wrong. They might discover that the real need is actually different from what they thought, which then leads to a very different solution.

There is no such thing as "requirements"; there are only unvalidated assumptions. The goal of delivering MMFs is to validate our assumptions as soon as possible, in case they turn out to be false and we need to change direction. Agile development assumes that the specific destination is unknown, so we need to iterate our way there. Software development is inherently a discovery process.

2. *Gamestorming*, p. 5

In knowledge work we need to imagine a world that doesn't exist yet and embrace exploration, experimentation, and trial and error. We are bringing fuzzy goals into reality. A fuzzy goal is one that "motivates the general direction of the work, without blinding the team to opportunities along the journey."[3] This is the journey we hope to demonstrate in this book, as the library development team adopts BDD and employs it to enable them to get better at delivering predictably in the face of fuzzy goals. "Fuzzy goals must give a team a sense of direction and purpose while leaving team members free to follow their intuition."[4]

Addressing Some Concerns

Sam and Jonah Discuss Sam's Concerns

Setting: Sam's office, right after the previous meeting. There are large, colorful, highly detailed business process diagrams plastered over the walls, a couple of comfy chairs, and the scent of fresh coffee.

JONAH: So after I ground the coffee I put it in the Aeropress and made my espresso. And the guy in the seat next to me said, "What on earth is that? Some kind of crazy airpump?"

SAM: *(Laughs)* Seriously? You made your own coffee? Right on the flight here?

JONAH: *(Laughs)* Sounds a little extreme, I know. I usually take the Aeropress when I travel, but that's actually the first time I've used it and the grinder on a plane. I'll probably try it on Friday's flight home—life's too short for in-flight service coffee. Plus it made the cabin smell so much better!

SAM: Ha! No question.

JONAH: Mmmm…This is great coffee, by the way. What is it?

3. *Gamestorming*, p. 5
4. *Gamestorming*, p. 5

SAM: Huehuetenango.

JONAH: "Way Way Tenango"? What is that, Ethiopian?

SAM: No, Guatemalan. It's spelled differently from how it sounds. My wife and I vacationed down there last summer and happened to tour through the area where it's grown. Guatemala was an amazing experience, I'll tell you more some other day.

Our day in the village was the highlight, though. We learned so much about the whole coffee process by talking with the people who actually grow it. We've been huge fans of Huehuetenango ever since. It tastes so good, and always brings back those same good memories.

JONAH: I bet. I'll be sure to track down some of my own. Thanks for the tip!

SAM: You're welcome. Anyway, about that last meeting…I might have come off a little strong in there. I don't want you to get the wrong idea, I'm actually very supportive of anything that can help us improve. I just don't like coming into a meeting as the only one not knowing what's going on.

JONAH: No worries. I figured that was the case. Once again, that was not my intention. I'll do everything I can to help get you up to speed.

SAM: Great. As a BA, I tend to work with all the development teams, and sometimes it can be a little hard to keep up with every new thing they're trying. We've had some good success with Scrum recently, but we've also had a lot of tools and initiatives die on the vine. It's hard to know ahead of time what's going to work and what's not. I'm very pragmatic, so I tend to adopt a wait-and-see posture with new things, to see if they deliver on the promise.

As you can see *(Gestures around)* I'm very process-oriented and always want to keep the bigger picture in mind, so I'm curious as to how this BDD stuff will fit into how the teams work. If this can help bring more structure and discipline to our process, I'm all in favor of it. Will it do that?

JONAH: I expect so. I'd be very surprised if it didn't.

SAM: So, I have some questions. First, what's a feature file?

JONAH: It's what we use to record our test scenarios.

SAM: What? Now I'm confused again. I thought you, Raj, and Robin were going to be writing up the scenarios we captured in this morning's meeting; didn't you say you would be "formalizing them" or something like that?

JONAH: I see. Let me try to clear up some confusion here. Yes, after lunch Raj and Robin and I will do some work formalizing the scenarios we came up with this morning. We'll write them up in what we call a "feature file," which is simply a plain-text description of how we expect the feature to work.

We want to do this formalization to clear up any ambiguities in our language, make sure we captured all the important details, and help us ensure we're all on the same page with you, Mark, and Jane about how it should work. The whole team, actually. It also lets us see if we missed anything.

SAM: Is this where Raj's vegetable makes an appearance? The "cucumber?"

JONAH: Correct! Though cucumbers, like tomatoes, are actually fruit. But that's not important right now.

Anyway, I got us a little off track there. Let's try that again.

There's a software tool called Cucumber that Raj, Robin, and Jane will use in development to treat our scenarios as a kind of "living specification" for the search feature. With Cucumber we'll be able to verify the search feature as they write it. In other words, Raj and Robin will code to the scenarios we create, one by one, running them as tests as they do. The feature file is what Cucumber uses as the place to record the scenarios we are using to specify what the feature should do for this story.

SAM: So this "feature file" is a specification document? We're all in a room just writing a specification document? This is insane. I could do this by myself. We don't need a meeting with the whole team.

Why do we need any meetings for this? Isn't this the kind of thing I would normally do on my own as a business analyst? Actually, on Jessie's team, doesn't someone write all these details in the product backlog item anyway?

JONAH: Good questions. The last thing we want to do is take up everyone's valuable time with useless meetings. We are having more meetings

than usual at the moment, since we're getting everyone on the same page with the new techniques, but don't think of BDD as a series of meetings. It's more like something a team just does in the course of development. In other words, in the course of the team doing their work, they collaborate; the right people get together whenever they see the need to figure out details.

SAM: OK, good. I have to admit, the meeting we had earlier today was not like many of the boring meetings we have. Everyone was pretty engaged, and I thought we had some pretty important insights and caught some important cases. I could see how it would be helpful to have the developers and testers as part of that conversation.

JONAH: Exactly. And as you'd expect, I'd love to see you all get to the point where you wouldn't even refer to this sort of thing as "having a meeting," since with BDD if you're describing it as a "meeting" it's usually an indication that something's wrong with your collaboration.

The goal for this morning's conversation was to get the right people in the room to discuss what the story meant and how we'll know it's done, and to make sure we don't miss any important details. It's also critical with a complex business domain like this to ensure the product owner, BAs, developers, and testers—the whole team—are all on the same page with the terminology, how the feature should work, and how it fits in with the rest of the application.

SAM: Yes, there is a lot of complexity in the library domain. More than people appreciate. And all the systems, applications, and integrations we have add a lot of technical complexity too. It's especially hard to build things on top of our legacy ILS.

JONAH: Ah, OK. So, yeah, I would expect that it's hard to build new features on your legacy ILS. Most of the teams I work with are doing BDD over the top of some kind of legacy system. It's a challenge, but it's not a unique one. Or, as I tell people, "Yes, you are a special snowflake, but not in this area."

SAM: *(Laughs)*

JONAH: Jessie's background in UXD is also very helpful in understanding the flow of user actions and moving beyond just thinking about the role

without considering the person using the software and the actual needs the new software feature is intended to meet. I really value having someone who can bring that perspective to the table.

So yes, a feature file is a kind of specification documentation. More like an executable specification. At least, that's what we're aiming for.

SAM: I'm not sure what you mean by "executable specification."

JONAH: In a traditional kind of approach often a product owner, or business analyst such as yourself, would be documenting these scenarios as acceptance criteria, and a tester would perhaps be using them as the basis for their test plan. The acceptance criteria would be captured in the product backlog as stories and their details, and then the test plans and other information would likely be separate supporting documents.

SAM: That's pretty much how we do it now.

JONAH: Right, and much of that part of the process will not change. However, as you can see, feature files are more dynamic than that. They're meant to be *more* than just a document.

With BDD we're aiming for a *living* document, one that grows and changes as the software matures.

Because the scenarios run as tests, they end up automating much of what would normally be recorded in manual test plans and regression tests, and because the developers code *to* the scenarios with Jane's early input, there is less chance of missing things and introducing bugs.

SAM: OK, but won't the feature file have to be constantly changing, such as when we update the UI?

JONAH: If we put a lot of UI-specific language in the feature file, then yes, it certainly would. Many teams fall into that trap by referring to "Submit buttons" and the like. But we won't. We'll take the time to keep only business language in the feature file. That's one area where your input will be invaluable. Help keep us honest!

Teams who do a good job at keeping the feature file language focused on business concerns find the scenarios remain much the same over time. The application's UI may change, and the underlying libraries and applications may too, but the Cucumber scenarios would change only

when the actual features change, since they describe the *behavior* of the application in business language, not the implementation. That's a lot to think about. Hopefully, I'm not jumping ahead too much.

SAM: A little. But I'm actually more concerned about where I fit in.

JONAH: Of course. Let's talk about the more important concern, which is *your* role in this kind of approach.

Let's see. Mark understands the big goals, your business goals. But he depends on your understanding of the nuances of business processes, systems, and your domain—all the stuff you're really good at—which is all the stuff that helps the team make sure we have all the right examples for each feature.

SAM: Yes, he does. Mark realizes that no one knows the business processes, technical ecosystem, and even all the library acronyms better than I do. In fact, Mark and Susan kid that after a few beers I might say I understand the library processes better than the librarians do. *(Grins)* But even I wouldn't go that far.

JONAH: *(Laughs)* In terms of what we're doing this week, this first meeting focused on the happy path, which wasn't really the part where you add the most value. But the next few conversations are where we've got the happy path down, and we'll be going through all the interesting variations. These will be where you're really going to contribute a lot more to this, since it's the part that's really going to depend on you and what you know.

SAM: This first story seems relatively straightforward to me, which is why I'm having some trouble understanding why we're taking so much time on it. At least, it's simple compared to what we'll have to do for tracking what ebooks each library patron is borrowing.

JONAH: Agreed. This first story is less involved when compared to what is coming. Part of that is deliberate, to help your team get up and running with BDD. To start practicing the skills with a simpler example so it's achievable. We'll move to more involved ones once you all have the basic skills down.

SAM: That makes sense. Sometimes I'm the one writing the stories, and sometimes Mark does that. Jessie pitches in, too, from time to time.

Will Mark and I be writing the feature files from now on, or will Raj and Robin do that?

JONAH: There's nothing that says a certain role has to be the one that ends up documenting the scenarios. It's an important process question, so I recommend we hold off on some questions like that for now and see what the team figures out.

Like learning any new team skill, a lot of this is going to be harder to explain ahead of time than just to learn it together by doing it. I'll do my best to answer any questions that come up, and I do really understand how you may be skeptical about trying this approach. I certainly would be if I was in your place. Since you trust the team, I'm sure you can also talk with any of them if you have more concerns.

Why not give it a go, at least for the next few sprints, and see how it works out?

SAM: *(Shrugs and nods)* OK. I'm willing to give it a try for now, support it, and see what happens.

JONAH: Excellent. Thanks again for the brew! See you after lunch.

Treat Resistance as a Resource

If you're reading this book, you're likely to be an advocate for BDD on your team, which means you're likely to run into resistance from other people who aren't as excited about the change.

You might see that resistance as something you need to fight against to successfully adopt BDD on your team. Or you might write it off as just "resistance to change." We'd like to suggest an alternative: Gratefully accept that resistance as a useful resource.[5]

For the most part, people don't actually resist change per se. People make changes all the time—and that person you think of as "resistant to change" would eagerly change many things in their life if they were to win the lottery. But people do resist particular changes, changes where they don't, for whatever reason, see a likely net positive outcome. This means that when you encounter resistance, you have an opportunity to learn something that might improve your proposed change. When

5. Emery, "Resistance as a Resource"

someone resists your proposed change, ask yourself, "What do they know that I don't know?"

To answer that question, we've found it useful to think in terms of different layers of resistance, or layers of buy-in, based on a model from Eli Goldratt's Theory of Constraints. There are several different formulations of this, with different numbers of layers, but we like Dr. K. J. Youngman's:

1. We don't agree about the extent or nature of the problem.
2. We don't agree about the direction or completeness of the solution.
3. We can see additional negative outcomes.
4. We can see real obstacles.
5. We doubt the collaboration of others.[6]

Start at the beginning of the list and look for where the resistance begins. Maybe the person agrees there's a problem to solve but they're not convinced your proposal actually solves it. Find out what they know about the solution; perhaps they've seen something similar in the past that didn't work. You might be able to learn something from that failure. Or, you might be able to persuade them that this solution is different.

Maybe they agree the solution will work but they also see potential side effects. Again, what do they know that you don't? Perhaps you need to add something to your proposed change to mitigate the side effects.

In the previous conversation, Jonah engaged Sam to explore Sam's resistance. Sam had three main objections, all at level 3:

- BDD will cause us to spend too much time in meetings.
- Feature files will have to change too often.
- My role will be marginalized or unappreciated in this new approach.

Notice that Sam wasn't objecting that things were fine and there was no need to change (level 1) or that BDD wouldn't solve their problems (level 2). He was saying that, even if it worked, BDD would cause negative side effects. So, Jonah engaged Sam in conversation about those potential side effects and how to prevent or mitigate them. Had Jonah focused on the problem and how BDD would solve it, he wouldn't have

6. Youngman, http://www.dbrmfg.co.nz/Bottom%20Line%20Agreement%20to%20Change.htm

won Sam's willingness to participate in the experiment. Sam might even have worked against the experiment. But because Jonah heard and engaged Sam's concerns, Sam's on board and his feedback can help make the experiment stronger.

One pleasant surprise with this approach to resistance is how often the person putting up the biggest resistance becomes the biggest supporter of the change once you listen to them and incorporate what they know.

Playing the BDD Game

Jonah introduced the idea that BDD, like Agile, is a cooperative game. Let's dig in to some of the practical implications of this as we think about exploring examples. The book *Gamestorming* presents the idea that every game has a common shape. This shape has three different stages, and each stage has a different purpose. This shape looks like that shown in Figure 2-3.

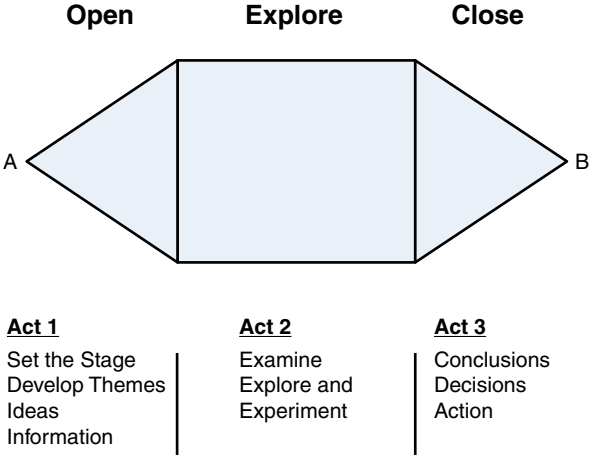

Figure 2-3 *Game Design*

In a game, a team wants to get from their known point A to fuzzy goal B, but they don't know how. So they apply this game framework to get there. We believe BDD includes this kind of game, with the same structure and accompanying mindsets. BDD is so much more than just this collaborative game, but this aspect of BDD is typically the hardest part for most teams to understand and master. Let's look at the three stages of the game structure.

Opening

The opening phase of the game is all about opening people's minds, opening up possibilities and ideas for later exploration. The opening is *divergent*, as illustrated in Figure 2-4. Generating new ideas is maximized and all critique is deliberately set aside. It's about getting as many ideas out in the open as possible and avoiding critical thinking and skepticism.

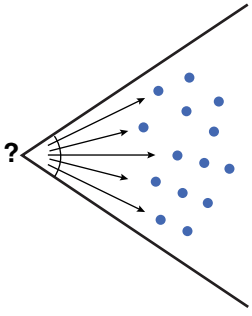

Figure 2-4 *Opening (Divergent)*

The opening stage in BDD involves brainstorming examples that help the team understand the business domain. These examples are focused on the customer experience and are as close as possible to the reality of the business domain. The goal is to generate a variety of examples that help the team understand the business domain and the customer need they are trying to address. Some teams split into pairs or triads to maximize the diversity of perspectives and ideas. This stage may take only a few minutes or much longer, depending on the complexity of the domain being considered.

Exploring

The keyword for the exploring stage is *emergent*. Exploration and experimentation are the focus. You want to create the conditions where "unexpected, surprising, and delightful things" emerge.[7] Figure 2-5 illustrates the nonlinear, emergent nature of the exploring stage.

In BDD, this stage builds on the energy and ideas that flowed into the room during the previous divergent stage, exploring the examples generated to see patterns and missing concepts and scenarios. If the team split into subgroups, this is when

7. *Gamestorming*, p. 11

the subgroups each take turns presenting their findings to the rest of the team, then the team looks for patterns, misunderstandings, missing concepts, and themes in the examples.

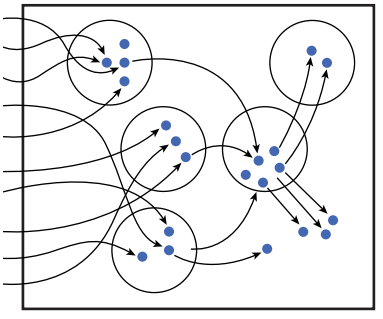

Figure 2-5 *Exploring (Emergent)*

The exploring stage can feel chaotic, directionless, and confusing to those not used to it. Thus, this stage can be very uncomfortable for teams that aren't used to exploring in this way. One facilitation book, the *Facilitator's Guide to Participatory Decision Making*, even refers to the exploring stage as "The Groan Zone," because the creative tension necessary for this stage takes effort to maintain and is discomforting for those not used to it.

Because of this tension, in the exploration stage of the game the temptation to critique options and converge on solutions as soon as possible can be very strong. If this happens, it can mean an early death to creativity, but this "groan zone" is a vital, normal, and necessary part of the creative process. Sam is used to formalizing proposed solutions early, which makes the exploration stage a big part of why he felt so uncomfortable playing the BDD game.

The right thing to do in exploration is to keep the creative tension and suspend judgment as long as necessary. This enables a team to pursue full participation, mutual understanding, inclusive solutions, and shared responsibility. We saw Jonah do this with the team, supporting and encouraging active dialogue about the various scenarios while not being afraid to dig a little deeper when necessary.

As we saw with the team, being very concrete is critical at this stage: who the user is in terms of background and experience, what they're trying to accomplish, what struggles they might have in getting their need met, where they are, and so on. All these kinds of details might seem quite incidental and unimportant, but they are

vital in helping everyone visualize each scenario and identify what's missing, which then helps the team see other scenarios.

The focus in the exploration stage is on integrating the various ideas and perspectives rather than critiquing them. This is where the whole is greater than the sum of the parts. The team may analyze certain examples and discard them, or at least postpone further discussion on them. They may discover other examples that illuminate the domain more, and thus are pursued further. The team talks together about each example to make sure they understand it, filling in missing pieces and making note of things requiring further investigation.

Closing

The closing stage is where a game converges on conclusions, decisions, and action. It's finally the time to bring in critical thinking and rigor, evaluating which ideas are the most promising to pursue and worth an investment of time and energy. The keyword for this stage is *convergent*, as illustrated in Figure 2-6. It's about narrowing the field "in order to select the most promising things for whatever comes next."

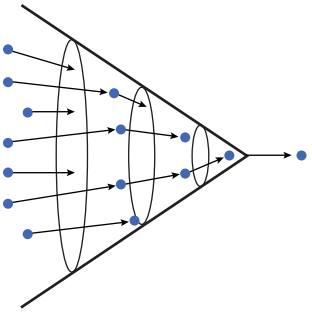

Figure 2-6 *Closing (Convergent)*

For the BDD "game," this means converging on a decision about which examples to carry through the rest of the BDD process. It involves starting to formalize the scenarios, looking for which details are significant and which are incidental. This leads us naturally to returning to our team in Chapter 3.

As we stated earlier, playing the BDD game is fundamentally about intentional discovery and growing a shared understanding. The overall BDD game structure looks like that shown in Figure 2-7, with divergent, emergent, and convergent stages.

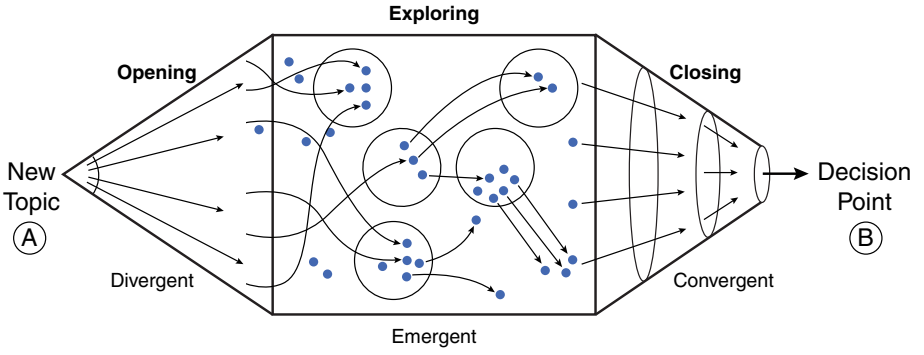

Figure 2-7 *BDD game structure*

Summary

- BDD is a collaborative game of exploring the desired behavior of a software system.
- People with different roles—usually product owner, developer, and tester—use examples to emerge this behavior rather than one individual trying to specify the behavior up front in isolation.
- Example mapping is a good technique for structuring a discussion around examples.
- Use real, concrete examples, not unrealistic dummy data.
- Use a common or happy path example to get to work on a real, concrete thing right away. Come back later to handle variations rather than trying to fully specify a story up front.
- Not everyone will be on board with trying a new way of working. Treat resistance as a resource, a source of information to engage rather than something to fight or avoid.
- Games like BDD have a natural structure—opening, exploring, and closing. Don't rush through the stages.

References

Cockburn, Alistair. *Agile Software Development: The Cooperative Game*. Boston: Pearson Education, Inc., 2007.

Emery, Dale H. "Resistance as a Resource." http://dhemery.com/articles/resistance_as_a_resource/

Gray, David, Sunni Brown, and James Macanufo. *Gamestorming: A Playbook for Innovators, Rulebreakers, and Changemakers*. Sebastopol, CA: O'Reilly, 2010.

Wynne, Matt. "Example Mapping": https://docs.cucumber.io/bdd/example-mapping/

Youngman, Dr. K. J. http://www.dbrmfg.co.nz/Bottom%20Line%20Agreement%20to%20Change.htm

Index

Symbols

A

B

C

D

E

F

G

H–I

J–K

L

M

N–O

P

Q

R

S

T

U

V

W

X–Y–Z