

GOJKO ADZIC



SAMPLE CHAPTER

SPECIFICATION BY EXAMPLE

How successful teams deliver the *right* software



MANNING



SPECIFICATION BY EXAMPLE

How successful teams deliver the right software

Gojko Adzic



Specification by Example
by Gojko Adzic

Chapter 1

Copyright 2011 Manning Publications

Brief Contents

Preface xiii
Acknowledgments xxii
About the author xxiii
About the cover illustration xxiv

- 1** Key benefits 3
 - 2** Key process patterns 17
 - 3** Living documentation 29
 - 4** Initiating the changes 36
 - 5** Deriving scope from goals 65
 - 6** Specifying collaboratively 77
 - 7** Illustrating using examples 95
 - 8** Refining the specification 114
 - 9** Automating validation without changing specifications 136
 - 10** Validating frequently 162
 - 11** Evolving a documentation system 183
 - 12** uSwitch 201
 - 13** RainStor 211
 - 14** Iowa Student Loan 217
 - 15** Sabre Airline Solutions 224
 - 16** ePlan Services 230
 - 17** Songkick 237
 - 18** Concluding thoughts 245
- Appendix A Resources 250
Index 255



Key benefits

In the internet age, delivery speed is the theme of the day in software development. A decade ago, projects lasted several years and project phases were measured in months. Today, most teams' projects are measured in months and project phases are reduced to weeks or even days. Anything that requires long-term planning is dropped, including big up-front software designs and detailed requirements analysis. Tasks that require more time than an average project phase are no longer viable. Good-bye code freezes and weeks of manual regression testing!

With such a high frequency of change, documentation quickly gets outdated. Detailed specifications and test plans require too much effort to keep current and are considered wasteful. People who relied on them for their day-to-day work, such as business analysts or testers, often become confused about what to do in this new world of weekly iterations. Software developers who thought they weren't affected by the lack of paper documents waste time on rework and maintaining functionality that's not required. Instead of spending time building big plans, they waste weeks polishing the wrong product.

In the last decade, the software development community has strived to build software the "right" way, focusing on technical practices and ideas to ensure high-quality results. But *building the product right* and *building the right product* are two different things. We need to do both in order to succeed.

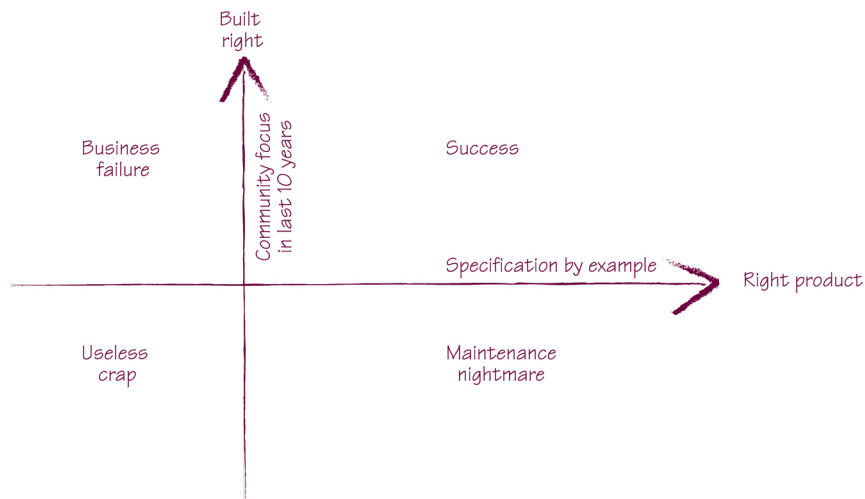


Figure 1.1 Specification by Example helps teams build the right software product, complementing technical practices that ensure that the product is built right.

To build the right product effectively, software development practices have to provide the following:

- Assurance that all stakeholders and delivery team members understand what needs to be delivered in the same way.
- Precise specifications so delivery teams avoid wasteful rework caused by ambiguities and functional gaps.
- An objective means to measure when a piece of work is complete.
- Documentation to facilitate change, in terms of both software features and team structure.

Traditionally, building the right product required big functional specifications, documentation, and long testing phases. Today, in the world of weekly software deliveries, this doesn't work. We need a solution that gives us a way to

- Avoid wasteful over-specifying; avoid spending time on details that will change before a piece of work is developed.
- Have reliable documentation that explains what the system does so we can change it easily.
- Efficiently check that a system does what the specifications say.
- Keep documentation relevant and reliable with minimal maintenance costs.
- Fit all this into short iterations and flow-based processes, so that the information on upcoming work is produced just-in-time.

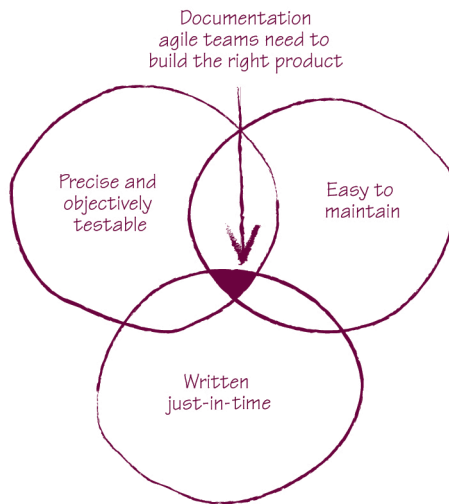


Figure 1.2 Key factors for the right kind of documentation for agile projects

Although these goals might seem in conflict at first, many teams have succeeded at fulfilling all of them. While researching this book, I interviewed 30 teams that implemented around 50 projects. I looked for patterns and common practices and identified underlying principles behind these practices. The common ideas from these projects define a good way to build the right software: *Specification by Example*.

Specification by Example is a set of process patterns that helps teams build the right software product. With Specification by Example, teams write just enough documentation to facilitate change effectively in short iterations or in flow-based development.

The key process patterns of Specification by Example are introduced in the next chapter. In this chapter, I'll explain the benefits of Specification by Example. I'll do so using Specification by Example style; instead of building a case for this book in a theoretical introduction, I'll present 18 real-world examples of teams that got big dividends from Specification by Example.

Before I begin, let me emphasize that it's hard to isolate the impact or effect of any single idea on a project. The practices described in this book work with—and enhance—the effectiveness of other, more established agile software development practices (such as test-driven development [TDD], continuous integration, and planning with user stories). When considering a range of projects in different contexts, patterns emerge. Some of the teams I interviewed were using an agile process before implementing Specification by Example, and some implemented Specification by Example while transitioning to an agile process. Most of the teams used iteration-based processes, such as Scrum and Extreme Programming (XP), or flow-based processes, such as Kanban—but some even used these practices in an environment that wouldn't be considered agile by any standard. Yet most reported similar benefits:

- *Implementing changes more efficiently*—They had living documentation—a reliable source of information on system functionality—which enabled them to analyze the impact of potential changes and share knowledge effectively.
- *Higher product quality*—They defined expectations clearly and made the validation process efficient.
- *Less rework*—They collaborated better on specifications and ensured a shared understanding of the expectations by all team members.
- *Better alignment of the activities of different roles on a project*—Improved collaboration led to a more regular flow of delivery.

In the next four sections, we'll take a closer look at each of these benefits using real-world examples.

Implementing changes more efficiently

In the course of researching this book, the most important lesson I learned concerned the long-term benefits of *living documentation*—in fact, I consider it one of this book's most important messages, and I cover it extensively. Living documentation is a source of information about system functionality that's as reliable as programming language code but much easier to access and understand. Living documentation allows teams to collaboratively analyze the impact of proposed changes and discuss potential solutions. It also allows them to make use of existing documentation by extending it for new requirements. This makes specifying and implementing changes more efficient over time. The most successful teams discovered the long-term benefit of living documentation as a result of implementing Specification by Example.

The Iowa Student Loan Liquidity Corporation, based in West Des Moines, Iowa, went through a fairly significant business model change in 2009. The financial market turmoil during the previous year made it nearly impossible for lenders to find funding sources for private student loans. Because of this, many lenders were forced to leave the private student loan market or change their business models. Iowa Student Loan was able to adapt. Instead of using bond proceeds to fund private student loans, it pooled funds from banks and other financial institutions.

In order to adapt effectively, they had to perform a “dramatic overhaul of a core piece of the system,” according to software analyst and developer Tim Andersen. The team used living documentation as a primary mechanism for documenting business requirements when they were developing their software. The living documentation system made it possible for them to ascertain the impact of new requirements, specify required changes, and ensure that the rest of the system works as it had before. They were able

to implement fundamental change to the system and release it to production in only one month. A living documentation system was essential for this change. Andersen said,

“Any system that didn’t have the tests [living documentation] would halt the development and it would have been a re-write.”

The Talia project team at Pyxis Technologies in Montreal, Quebec, had a similar experience. Talia is a virtual assistant for enterprise systems, a chat robot with complex rules that communicates with employees. From the first day of development, the Talia team used Specification by Example to build a living documentation system. A year later, they had to rewrite the core of the virtual agent engine from scratch—and that’s when the investment in living documentation paid off. André Brissette, the Talia product director, commented,

“Without that, any major refactoring would be a suicide.”

Their living documentation system made the team confident that the new system would work the same as the old one when the change was complete. It also enabled Brissette to manage and track the project’s progress.

The team at Songkick, a London-based consumer website about live music, used a living documentation system to facilitate change when redeveloping activity feeds on their site. They had realized that the feeds were implemented in a way that wouldn’t scale to the required capacity; living documentation supported them when they were rebuilding the feeds. Phil Cowans, the CTO of Songkick, estimates that the team saved at least 50% of the time needed to implement change because they had a living documentation system. According to Cowans,

“Because we had such a good coverage and we really trusted the tests [in the living documentation system], we felt very confident making big changes to the infrastructure rapidly. We knew that the functionality wouldn’t change, or if it did change, it would be picked up by a test.”

The development team at ePlan Services, a pension service provider based in Denver, Colorado, has used Specification by Example since 2003. They build and maintain a financial services application with numerous stakeholders, complex business rules, and complex compliance requirements. Three years after starting the project, a manager with unique knowledge about the legacy parts of the system moved to India. According to Lisa Crispin, a tester working for ePlan Services and author of *Agile Testing: A Practical*

Guide for Testers and Teams (Addison Wesley, 2009), the team worked hard to learn what the manager knew and build it into living documentation. A living documentation system enabled them to capture the specialist’s knowledge about their business processes and make it instantly available to all the team members. They eliminated a bottleneck in knowledge transfer, which enabled them to efficiently support and extend the application.

The Central Patient Administration project team at the IHC Group in Oostkamp, Belgium, implemented a living documentation system with similar results. The ongoing project, which started as a rewrite of a legacy mainframe system, began in 2000. Pascal Mestdach, a solution architect on the project, said that the team benefited greatly:

“There were just a few people who knew what some functionality on the legacy system did—that became much clearer now because the team has a growing suite of tests [living documentation] against that functionality and it describes what it does. Also questions can be answered when a specialist is on holiday. It’s more clear to other developers what a piece of software is doing. And it is tested.”

These examples illustrate how a living documentation system helps delivery teams share knowledge and deal with staff changes. It also enables businesses to react to market changes more efficiently. I explain this in more detail in chapter 3.

Higher product quality

Specification by Example improves collaboration between delivery team members, facilitates better engagement with business users, and provides clear objective targets for delivery—leading to big improvement in product quality.

Two case studies stand out. Wes Williams, an agile coach from Sabre Holdings, and Andrew Jackman, a consultant developer who worked on a project at BNP Paribas, described how projects that had failed several times before succeeded with Specification by Example. The approach described in this book helped their teams conquer the complexity of business domains that were previously unmanageable and ensure high quality of deliveries.

At Sabre Holdings, Wes Williams worked on a two-year airline flight-booking project complicated by global distribution and data-driven processes. The project involved 30 developers working in three teams on two continents. According to Williams, the first two attempts to build the system failed, but the third—which used Specification by Example—succeeded. Williams had this to say:

“We went live with a large customer [a big airline] with very few issues and had only one severity 1 issue during [business acceptance] testing, related to fail-over.”

Williams estimates that Specification by Example was one of the keys to success. In addition to ensuring higher quality, Specification by Example also facilitated trust between developers and testers.

At BNP Paribas, the Sierra project is another great example of how Specification by Example leads to high-quality products. Sierra is a data repository for bonds that consolidates information from several internal systems, rating agencies, and other external sources and distributes it to various systems inside the bank. Various systems and organizations used the same terms with different meanings, which caused a lot of misunderstanding. The first two attempts to implement the system failed, according to Channing Walton, one of the developers on the team that helped make the third attempt a success. The third effort succeeded partially because Specification by Example enabled the team to tackle complexity and ensure a shared understanding. Product quality of the end result was impressive. The project has been live since 2005 “with no major incidents in production,” according to Andrew Jackman, a consultant developer on the Sierra project. Most people currently working on the Sierra project were not there when the project started, but the level of quality is still very high.

Similar results were obtained by Bekk Consulting for a branch of a major French bank with a car-leasing system. According to Aslak Hellesøy, a member of the original team and the author of Cucumber, a popular automation tool that supports Specification by Example, they had only five bugs reported in the two years since the system went live, even though the software is now maintained by a completely new team.

Lance Walton worked as a process consultant for a branch of a large Swiss bank in London on a project to develop an order-management system that had failed to start several times before. Walton stated that the project was implemented in an environment where it was assumed that systems required a support team at least as big as the development team. His team used Specification by Example and delivered a system to production nine months after the project started, passed the business acceptance testing in one day, and reported no bugs for six months after that. According to Walton, the new system required no additional support staff, cost less than predicted, and enabled the team to deliver a finished product earlier. In comparison, the team next to them had ten times more people working on support than development. According to Walton,

“At the moment the team is still releasing every week and the users are always happy with it. From the point of quality, it is superb.”

The techniques of Specification by Example work for brownfield as well as greenfield projects. It takes time to build up trusted documentation and clean up legacy systems, but teams see many benefits quickly, including confidence in new deliverables.

A good example is the foreign exchange cash-management system at JP Morgan Chase in London. Martin Jackson, a test automation consultant on the project, said that the business analysts expected the project to be late—instead, it was delivered two weeks early. High product quality enabled them to successfully complete the business-acceptance testing phase in a week instead of four weeks, as originally planned. Jackson said,

“We deployed it and it worked. The business reported back to the board as the best UAT experience they ever had.”

Specification by Example also enabled Jackson’s team to quickly implement “quite a significant technical change” late in the project development, improving the precision of calculations. Jackson reported:

“All the functionality covered by the FitNesse suite [living documentation] went through the whole of system test, whole of UAT, and live to production without a single defect. There were several errors outside of the core calculation components that were captured during system testing. What made the UAT experience so good for the business was that when calculation errors appeared, we were all pretty certain that the root cause was going to be upstream from the calculation code itself. As a result of the FitNesse suite, it was easier to diagnose the source of defects and hence the cleaner and faster delivery through to production.”

The software development team at Weyerhaeuser in Denver, Colorado, writes and maintains several engineering applications and a calculation engine for wooden frames. Before applying Specification by Example, construction engineers were not usually involved in software development, even though the team was dealing with complex scientific calculation formulas and rules. This caused numerous quality issues and delays, and the process was further complicated by the fact that the engine is used by several applications. According to Pierre Veragen, the SQA lead on the project, the hardening phase prior to release would drag on and a release would rarely go out without problems.

After implementing Specification by Example, the team now collaborates on specifications with structural engineers and automates the resulting validations. When a change request comes in, the testers work with structural engineers to capture the expected calculation results and record them as specifications with examples before development begins. The engineer who approves a change later writes the specifications and tests.

Veragen states that the main benefit of the new approach is that they can make changes with confidence. In early 2010, with more than 30,000 checks in their living documentation system, they haven't noticed big bugs in years and have now stopped tracking bugs. According to Veragen:

“We don't need the [bug count] metrics because we know it's not coming back...engineers love the test-first approach and the fact that they have direct access to automated tests.”

Lance Walton worked on a credit risk-management application for a branch of a large French bank in London. The project began with external consultants helping the team adopt Extreme Programming (XP) practices, but they did not adopt any of the Specification by Example ideas (although XP includes customer tests, which is closely related to executable specifications). After six months, Walton joined the project and found the quality of the code to be low. Although the team was delivering every two weeks, the code was written in a way that made validation complicated. Developers tested only the most recently implemented features; as the system grew, this approach became inadequate. “When a release happened, people would sit around nervously, making sure that everything was still running and we'd expect a few issues to come up within hours,” said Walton. After they implemented Specification by Example, the quality and confidence in the product significantly improved. He added:

“We were pretty confident that we could release without any issues. We got to the point where we would quite happily deploy and go out for lunch without sticking around to see if it was OK.”

In contrast, a website-rewrite project at the Trader Media Group in the United Kingdom suffered from quality problems when the team stopped using Specification by Example. Initially, the team was collaborating on specifications and automating the validation. They stopped under management pressure to deliver more functionality earlier and faster. “We noticed that the quality took a nose dive,” said Stuart Taylor, the test team leader. “Where before it was quite hard for us [testers] to find defects, later we found that one story could produce four, five defects.”

Not only for agile teams

Collaborating on specifications isn't something that only agile teams can benefit from. In *Bridging the Communication Gap*,[†] I suggested that a similar set of practices could be applied to more traditional structured processes. After the book was published, I came across an example of a company that did just that while researching this book.

Matthew Steer, a senior test consultant at the Sopra Group in the UK, helped a major telecommunication company with a third-party offshore software delivery partner implement these practices. The main reason for change was the realization that projects were suffering from poorly defined requirements. Steer compared delivery in the year when ideas were implemented to the costs of delivering software the previous year. Not surprisingly, with a Waterfall approach these projects did not get to a zero-defect level, but the changes “increased upstream defect detection and reduced downstream rework and costs.” According to Steer:

“We were able to demonstrate the effectiveness of this approach by catching many more defects earlier in the life cycle that were traditionally found at later phases. The volumes of defects at the end of the life cycle significantly reduced and the pile increased at the early phases of the life cycle.”

The end result was a delivery cost savings of over 1.7 million GBP in 2007 alone.

[†] Gojko Adzic, *Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing* (Neuri Limited, 2009).

Less rework

Generally, frequent releases promote quick feedback, enabling development teams to find mistakes and fix them sooner. But iterating quickly doesn't prevent mistakes. Often, teams take three or four stabs at implementing a feature; developers claim this is because customers don't know what they want until they get something to play with. I disagree. With Specification by Example, teams generally hit the target in the first attempt. This saves a lot of time and makes the delivery process more predictable and reliable.

The Sky Network Services (SNS) group at British Sky Broadcasting Corporation in London is responsible for broadband and telephony provisioning software with high business workflow and integration complexity. The group consists of six teams. They have been using Specification by Example for several years. According to Rakesh Patel, a senior agile Java developer there, “We do tend to deliver when we say we do,” and

the group has a great reputation within Sky. At one time, Patel briefly worked with a different organization; he compared the two teams as follows:

“Every time they [developers in the other organization] give software to testers towards the end of the sprint, testers find something wrong and it always comes back to the developers. But here [at Sky] we don’t have that much churn. If we have an issue, we have an issue to make a test go green during development—it either does or it doesn’t. We can raise it there and then.”

Several other teams noticed a significant reduction of rework, including LeanDog, a group developing an aggregation application for a large insurance provider in the United States. Their application presents a unified user interface on top of a host of mainframe and web-based services and is further complicated by a large number of stakeholders from across the country. Initially, the project suffered from many functional gaps in requirements, according to Rob Park, an agile coach at LeanDog who helped the team with the transition. He said,

“As we started figuring stuff out, we needed clarification, and then we found out that we have to actually do something else.”

The team implemented Specification by Example, which resulted in much better specifications and reduced rework. Although developers continue to have questions for business analysts when they start working on a story card, “The questions have dropped considerably, as has the amount of back and forth we have to have and the questions are a lot different,” said Park. For him, the most rewarding aspects of Specification by Example is “getting the sense of the story and knowing the extent of the story as you start to build it.”

Many teams have also discovered that using Specification by Example to make requirements more precise at the start of a development cycle makes it easier to manage product backlogs. For example, being able to spot stories that are too vague or have too many gaps in required functionality early on prevents problems later. Without Specification by Example, teams often discover problems in the middle of an iteration, which interrupts the flow and requires time-consuming renegotiations—in larger companies, stakeholders who decide on the scope are often not readily available.

Specification by Example helps teams establish a collaborative specification process that lowers problems in the middle of an iteration. Additionally, Specification by Example fits into short iterations and doesn’t require months of writing long documents.

Less rework is a major advantage for the Global Talent Management team at Ultimate Software in Weston, Florida. Collaborating on specifications had a significant impact on focusing the development effort. According to Scott Berger, a senior development engineer in test at Ultimate Software:

“Meeting with our product owners to review our test scenarios prior to the team accepting a story readily allows the working group (product owner, developer, tester) to clarify ambiguous or missing requirements. On occasion, the outcome of the meeting has even resulted in the cancellation of stories, for example, when test scenarios reveal hidden complexity or conflicting requirements within the system. After one such discussion, the decision was made to nearly redesign an entire feature! Product owners are afforded the opportunity to rewrite and reslice the specifications, as opposed to having the development effort begin and halt or cancel the story midstream. By holding these meetings, we find ourselves being both more productive and efficient, because waste is reduced and vague and missing specifications are minimized. It also allows the team to come to a common understanding of what is expected.”

Most teams have significantly reduced or completely eliminated rework that occurred as a result of misunderstood requirements or neglected customer expectations. The practices described in this book allowed teams to engage better with their business users and ensure a shared understanding of results.

Better work alignment

Another important benefit of Specification by Example is the capacity to align different software development activities into short iterative cycles. From my experience and according to the case studies in this book, one of the most common stumbling points for teams moving to Scrum is the inability to fully complete tasks inside an iteration. Many teams hold onto the “old world” concepts: finish development first, then finish testing, and, finally, polish the product enough for it to be deployable. This fosters the illusion that development is completed in stages, when in fact subsequent testing and fixing are required for completion. One “done” column on the Scrum board means a developer thinks something is finished, a “done-done” column means the tester agrees, and so on (there are even reports of “done-done-done” columns). Work often falls into this pattern, and the results from testing affect the next cycle, causing much variability and making the delivery process less predictable.

Specification by Example resolves this issue. The practices described in this book enable teams to clearly define a target that’s universally understood and objectively measured. As a result, many teams find that their analysis, development, and testing activities became better aligned.

A good example of improved alignment occurred at uSwitch, one of the busiest websites in the United Kingdom. uSwitch implemented Specification by Example in 2008 because they had difficulty knowing when a feature was completed. “We’d finish something, give it over to the QA department, and they would immediately say to us that we forgot to test it in a certain scenario. This caused a lot of problems for us,” said Stephen Lloyd, a developer who works on the website. By implementing Specification by Example, they overcame that problem. Lloyd said that they’re now better integrated as a team and have a better understanding of the needs of the business. The process changes also resulted in improved software quality. Hemal Kuntawala, another developer working on the site, had this comment:

“Our error rates have dropped significantly, across the site. The turnaround of fixing problems is much quicker than it was previously. If a problem does occur on the live site, we can normally get a fix out within a few hours, where previously it took days or weeks to get something fixed.”

The team at Beazley, a specialist insurance company, also experienced improved alignment. Their business analysts work from the United States with developers and testers in the United Kingdom. They implemented Specification by Example primarily to ensure that software is finished when an iteration ends. Ian Cooper, a development team leader at Beazley, said:

“We’ve always done unit testing but the problem was that there was a gap around these tests telling us if the software works, not if it does what the customer wanted. We didn’t even use to have testers testing in the same cycle. They were feeding back the information from the previous iteration into the current iteration. That’s gone now. We have a much clearer idea of acceptance.”

The team working from New Zealand on AdScale.de, a marketplace for online advertising, had similar experiences. Two years after the project started, increasing complexity of the user interface and system integrations made the code base too large to be effectively managed just with unit testing. Developers would think that something was done, move on, and then have to redo the work after the testers’ review. Because of the disconnect between testers and developers, it took a long time to find problems. Issues from previous iterations were affecting future ones, disrupting the flow of development. After implementing Specification by Example, development and testing were more closely aligned. Clare McLennan, a developer/tester working on the project, declared:

“It took a lot of pressure from the release process—because the feedback is instantaneous. Previously, developers would be frustrated at us because their features hadn’t gone out. At the same time we were frustrated at them because they haven’t fixed the thing so we couldn’t test their features. We were waiting for them and they were waiting for us. That’s gone now because it only takes an hour to do all the testing. The features aren’t coming back into the next iteration.”

Specification by Example allows teams to define expected functionality in a clear, objective, and measurable way. It also speeds up feedback, improving the development flow and preventing interruptions to planned work.

Remember

- Building the product right and building the right product are two different things. You need to do both in order to succeed.
- Specification by Example provides just enough documentation at the right time, helping to build the right product with short iterations or flow-based development processes.
- Specification by Example helps to improve the quality of software products, significantly reduces rework, and enables teams to better align analysis, development, and testing activities.
- In the long term, Specification by Example helps teams create a living documentation system, a relevant and reliable description of the functionality that’s automatically updated with the programming language code.
- The practices of Specification by Example work best with short iterative (Scrum, Extreme Programming [XP]) or flow-based (Kanban) development methods. Some ideas are also applicable to structured development (Rational Unified Process, Waterfall) processes, and there have been cases where companies saved millions as a result.



“Unique, distilled knowledge from extensive industry research.”

—Mike Stockdale
Syterra Software

“I love this book. This is testing done right.”

—Craig Smith, Suncorp

“It will change the way we talk and think about testing.”

—David Evans
ThinkAlike Consulting

“The best book on requirement collection and maintenance.”

—Oleksandr Alesinsky
NAVTEQ

“Best book I have read in ages.”

—John Stevenson
Lean Agile Machine

“Based on the experience of many teams, it will double the value of your test automation.”

—Rick Mugridge, Rimu Research

SAMPLE CHAPTER

SPECIFICATION BY EXAMPLE

Gojko Adzic

Specification by Example is a collaborative method for specifying requirements and tests. Seven patterns, fully explored in this book, are key to making the method effective. The method has four main benefits: it produces living, reliable documentation; it defines expectations clearly and makes validation efficient; it reduces rework; and, above all, it assures delivery teams and business stakeholders that the software that's built is right for its purpose.

This book distills from the experience of leading teams worldwide effective ways to specify, test, and deliver software in short, iterative delivery cycles. Case studies in this book range from small web startups to large financial institutions, working in many processes including XP, Scrum, and Kanban.

This book is written for developers, testers, analysts, and business people working together to build great software. For additional resources go to specificationbyexample.com.

What's Inside

- Common process patterns
- How to avoid bad practices
- Fitting SBE in your process
- 50+ case studies

About the Author

A UK based consultant, **Gojko Adzic** helps teams worldwide implement Specification by Example and agile testing practices.

For access to the book's forum and a free ebook for owners of this book, go to manning.com/SpecificationbyExample

