Agile Business Analysis

Enabling Continuous Improvement of Requirements, Project Scope, and Agile Project Results

Kevin Aguanno
PMP, PMI-ACP, CSM

Ori Schibi
PMP, PMI-PBA, PMI-ACP, SMC

J.ROSS
PUBLISHING

ISBN-13: 978-1-60427-148-5

Printed and bound in the U.S.A. Printed on acid-free paper.

10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data
Names: Aguanno, Kevin, author. | Schibi, Ori, 1971- author.
Title: Agile business analysis : enabling continuous improvement of requirements, project scope, and agile project results / by Kevin Aguanno and Ori Schibi.
Description: Plantation, FL : J. Ross Publishing, [2018] | Includes index.
Identifiers: LCCN 2018025763 (print) | LCCN 2018034200 (ebook) | ISBN 9781604278019 (e-book) | ISBN 9781604271485 (hardcover : alk. paper)
Subjects: LCSH: Business analysts. | Business planning. | Agile software development.
Classification: LCC HD69.B87 (ebook) | LCC HD69.B87 .A48 2018 (print) | DDC 658.4/012—dc23
LC record available at https://lccn.loc.gov/2018025763

To all of those who have struggled through their
own agile transformations:

May your efforts inspire others to implement
changes in order to ensure that those who
follow you will find that their paths
forward have been smoothed.

CONTENTS

FOREWORD

From the moment an organization begins to adopt agile strategies, everyone asks a fundamental question: "How do I fit in?" This is an easy question to answer if you're currently a developer or tester, but it may be a little harder if you're a designer or architect, and could be rather difficult if you're a manager or business analyst.

It's completely natural to ask how you fit into agile and, frankly, I'd be concerned if you didn't ask that question. When I'm working with people who are trying to become agile, I recommend that they begin by exploring these questions: "How is agile different?" and "What is the agile mindset?" The bad news is that it will take you months, and perhaps even years, to truly internalize the answers to these questions. The good news is that you can easily start learning how to *be agile* if you're open to new (and proven) agile ideas. This new mindset includes a desire for close collaboration with both stakeholders and teammates; a desire to reduce the feedback cycle from the time that you hear an idea to the time it's implemented; a desire to reduce waste by focusing on effective communication and executable specifications over static specifications; and a desire to broaden your skill set beyond business analysis and to share your valuable analysis skills with others. Hopefully, you are doing some of these things already.

It isn't sufficient to just *be* agile, regardless of what some agile coaches may claim—you need to know how to *do* agile, too. This leads us to critical questions such as: "How does business analysis fit into agile?" "Who is doing this work?" and "What skills do I need to add value on agile teams?" This is where things get interesting for traditional business analysts. In order, the initial answers to these questions are: analysis is so important on agile teams that we do it throughout the entire life cycle, although we might not need to do it every single day; the person(s) with the skills and ability (who may or may not be an agile business analyst); and lightweight agile modeling and documentation skills, to start.

This leads us to the crucial question: "When would a business analyst be needed on an agile team?" The honest answer is: "Not always"—although perhaps a more palatable answer is: "Whenever the situation is sufficiently complex to warrant a specialized person in that role."

My experience, after working with dozens of agile teams over almost two decades on five continents, is that agile business analysts are needed on agile teams that are working at scale. This includes agile teams that are taking on complex domain problems, large teams (often organized as teams of teams), teams in life-critical or regulatory environments, and on geographically distributed teams. Agile methods such as Scrum don't address such scaling challenges well and don't include the role of business analyst as a result or, perhaps because Scrum doesn't include a business analyst role, it doesn't scale well in practice—you be the judge. Interestingly, the Large-Scale Scrum (LeSS) method, which is geared for teams of teams that are often geographically distributed, does include an explicit role of business analyst. The Disciplined Agile (DA) framework also suggests having business analyst specialists on teams when you run into any of the previously mentioned complexities.

So what's my point? The authors of this book answer these questions, and more, in detail. I believe that *Agile Business Analysis*: *Enabling Continuous Improvement of Requirements, Project Scope, and Agile Project Results* will prove to be a valuable resource on your learning journey. Read it now and take its advice seriously.

Scott Ambler
April 2018

Scott Ambler is an enterprise transformation coach and senior consulting partner at Scott Ambler + Associates (ScottAmbler.com). Scott led the development of the agile modeling (AM) (agilemodeling.com) and agile data (agiledata.org) methods in the early 2000s and is a co-creator of the DA framework (disciplinedagiledelivery.com).

INTRODUCTION

The decision to write this book came as a result of both of the authors' experience in agile project environments and their observations about the emerging trends and challenges in project management, business analysis, and the application of agile concepts. Kevin Aguanno's experience puts him at the forefront of adapting agile methodologies since he has been involved with these practices since before they were called agile. He was the first person to coin the term *agile project management*, and he has since helped shape the agile world by pioneering the concepts of disciplined agile delivery. Ori Schibi's experience includes introducing agile concepts in multiple settings and pragmatically adapting the approach to the specific needs of projects and the individual organizations. Ori is also a thought leader in the area of improving the collaboration between the project manager (PM) and the business analyst (BA).

Both authors' expertise extends beyond the application of agile concepts, and includes performing agile organizational readiness assessments that identify areas to improve and adjust in order to allow agile to succeed. This helps organizations choose the right approach (agile versus waterfall or something in between) for projects, and incorporating business analysis to support the successful delivery of agile projects.

Through their experiences, both authors noticed a growing tension in the background of agile projects; a struggle that reflects two growing and, at times, conflicting trends that seem to clash in the minds of many: agile and business analysis. Both agile and business analysis have grown in parallel, but the application of these areas takes place, for the most part, in different settings. Many organizations utilize BAs for both project and non-project work. The formation of the International Institute of Business Analysis (IIBA) in 2003 introduced a new era for the business analysis profession as it was the first attempt to define the role of the BA—with an enhanced focus on requirements. The Project Management Institute (PMI) followed through and introduced the Collect Requirements Process as part of Project Scope Management before further expanding the role of business analysis with the introduction of the PMI-PBA

Certification. The increased focus on business analysis further clarified the role of the BA and his/her involvement within projects and at the enterprise level.

With the rise of the BA role came obvious challenges, such as the blurry delineation of responsibilities between the PM and the BA. The lack of specific role definitions has led to friction, misunderstandings, gaps, and duplication of efforts as both worlds were colliding.

While the growth in business analysis was taking place, agile started to pick up, but in a parallel trajectory. Agile does not call specifically for BAs to be included in the projects and, in fact, the entire setup of agile projects appears to be designed to eliminate the need for intermediaries such as the BA. In the early days (after the introduction of the term agile and the *Agile Manifesto* in 2001) it appeared that there was no need for BAs. Early adaptations of agile were mostly confined to technology projects that were small. Team members were handpicked and the level of scrutiny was high. Over the years, the use of agile started to expand rapidly and become widely accepted. Agile has been scaling and growing—from small to all project sizes; from technology projects to pretty much any industry; and from more nimble, risk-taking, small, and change-hungry organizations to more traditional settings such as large financial institutions, governments, and other more conservative and slow-moving organizations.

It is no secret that even today many agile practitioners believe that there is no room or need for a BA on agile projects. On paper they may be right; however, there are two distinct drivers that point at agile projects' ability to benefit from BAs, or at least from the application of business analysis skills:

1. There are many areas in agile projects that have built-in challenges. These challenges can be covered and looked after with the application of business analysis skills. Many organizations struggle with getting clear mandates from their product owners; ownership issues around backlog maintenance; coordination between the business, testers, and developers; task updates; work allocation; and escalation. In well-functioning agile teams, nearly all team members should demonstrate basic business analysis skills; in reality, however, this may not be the case. Many teams struggle with the coordination of these issues and the true ownership of handoffs and other tasks that *fall* under the realm of business analysis.
2. The rapid and widespread adoption of agile has introduced new challenges that can benefit from business analysis skills. In the past, if agile team members were *handpicked* to ensure their skills and experience levels were applicable to the agile project's needs, the growth of agile ensures it becomes more difficult to guarantee that all team members are suitable for this kind of environment. Throughout this book, we establish that agile is a very unforgiving approach. There is a need for all

team members and stakeholders to *play along* and perform their roles, and for processes, policies, and governance structures to be aligned with an agile project's needs. When people or processes around agile projects are not aligned with the agile project, things will unravel quickly, causing project performance to lag. With the fast pace and the challenges that agile methods introduce, there is a consistently growing need for a BA to provide another set of eyes looking for gaps and picking up the pieces when things are misaligned. Although it is expected that business analysis activities are shared by all team members, in many agile environments, team members do not perform business analysis skills consistently, or even at all. Further, even when team members demonstrate business analysis activities, there are two types of challenges that surface: (1) when team members wear two hats (for example when their roles are split between being a developer and a BA), the BA role is secondary and when there are issues or time constraints, the secondary role typically gets pushed aside and (2) when multiple team members are responsible for business analysis activities, the seams and touchpoints between team members tend to become rough and inconsistent.

BOOK OBJECTIVES

This book examines the benefits that agile projects can gain by ensuring that there are business analysis skills present and that team members demonstrate them consistently. The book does not specifically call for a dedicated BA role in projects; however, it identifies many areas where a BA can complement the skills and experience of other team members and benefit the project. In fact, BAs can help support the product owner, the Scrum Master (or PM), and individual team members (ranging through all roles).

This book introduces considerations and criteria to determine whether it is sufficient to grow business analysis skills among team members or whether it is necessary to have an individual who has a BA title. The intent of the book is to advocate that whatever form is used, projects can benefit from business analysis skills. The book promotes the need to focus on bringing these two growing areas together—agile and business analysis—not by force but rather as a growing necessity.

Pride, ego, and misconceptions can often cloud people's judgment, and even the most knowledgeable and experienced individuals often get locked in paradigms and preconceived notions that prevent them from realizing opportunities. Many agile practitioners are adamant in their refusal to introduce BAs into agile projects, but an increasing number of practitioners realize the need for business analysis skills within their teams.

We believe that the biggest barrier is ensuring that there is recognition of the need for business analysis skills and, once acknowledged, there can be a healthy and productive discussion on how to best introduce these skills into agile projects. This book facilitates this discussion and helps decision makers reach the right conclusion based on the needs of the project, the organization, and the team. Such discussions, if done properly, incorporate all surrounding conditions, circumstances, constraints, and issues that agile teams—and managers—face.

WHAT MAKES THIS BOOK DIFFERENT

This book reviews agile concepts and looks at how organizations can benefit from agile delivery approaches, if applicable to their needs. The considerations on whether to use agile are viewed from a business analysis perspective. The book will help you to decide *if* and *where* business analysis concepts can be applied to help agile projects, and explores criteria determining whether the business analysis skills should be applied throughout the team or by introducing a BA to be part of the team.

The authors bring their unique styles and experiences with applying disciplined approaches to agile into the discussions around the need for business analysis skills. Their pragmatic approach does not try to impose ideas on the reader; rather, it focuses on enabling the reader to make the right decision as if the book is tailored to the readers' needs and circumstances.

THE STRUCTURE OF THE BOOK

Chapter 1 provides an introduction to agile and agile processes. It provides basic comparisons between waterfall and agile, continues with a review of agile concepts, and supports the review with a discussion of lean principles. Throughout the chapter, several ideas on how to minimize waste with the help of business analysis are introduced.

Chapter 2 discusses agile challenges. Agile provides many answers to the challenges introduced by Waterfall approaches, but agile is not perfect. First, it is important to recognize that agile has its own challenges; then it is time to review these challenges and learn how not to stumble on them—and if we do stumble, how to overcome these challenges. The primary challenges that this chapter covers include different views of what is agile, misconceptions and challenges related to culture, and organizational change. After a review of the *Agile Manifesto*, the chapter discusses how to deal with the organizational change elements as a result of agile adaptations, providing an approach for agile readiness and governance, and ensuring that the organization chooses the right approach. The chapter closes by reviewing additional challenges (and ways to overcome them) including focusing too much on the process, resource issues in a matrixed organization, the *diluting effect* that agile projects face due to rapid growth, and finally biases and challenges related to the Scrum method.

Chapter 3 examines roles and responsibilities in agile environments and shows the potential of the BA to help overcome these challenges. The chapter then moves to a more specific definition of the role of the BA in agile environments, what to expect when there is a BA assigned to agile projects, and role impact: how the BA supports and impacts the traditional agile roles of product owner, Scrum Master, and team members.

Chapter 4 reviews agile requirements processes, including the use of user stories. After an overview of the topic, the chapter takes the readers through a discussion of the relationship between user stories and use cases and how they support the agile requirements process. Beyond exploring use cases (both scenarios and diagrams) and the role of the BA in supporting their creation and maximizing the benefits they produce, the chapter reiterates the differences between *traditional* and agile requirements and shows how the BA can provide valuable support for the product owner. The book reviews different types of requirements to ensure no requirement *type* is overlooked and examines other requirements-related elements including themes, epics, and modeling.

Chapter 5 deals with agile documentation, a topic that is greatly misunderstood. Many people still believe *agile* means creating no documentation (in addition to no planning) and working fast, but these are all misconceptions. The chapter illustrates the importance of documentation in agile projects, shows how the BA plays an instrumental role in ensuring that the right documentation is created at the right level of detail, and looks into agile reporting, artifacts, project reports, and information radiators.

Chapter 6 focuses on the role of the BA in planning and estimating, which is different than the role a BA would fulfill in traditional waterfall projects. The BA supports the estimating and planning process by demonstrating the same skills he or she would in waterfall projects; however, the application of the skills in agile projects is different. This chapter also reviews business analysis activities that try to enhance the focus of the product owner and then analyzes the different planning cycles in agile projects and how they can benefit from a good business analysis—release planning, iteration planning, and daily planning. Other important concepts are also discussed through the lenses of how they can benefit from a BA: the prioritization process and techniques to estimate complexity and team velocity.

Chapter 7 provides an additional review of the prioritization process and looks for ways to improve processes around backlog maintenance, end of iteration demos, retrospectives, and applying lessons learned to upcoming iterations.

Chapter 8 covers the area of agile testing and solution evaluation. With testing often becoming an issue in agile projects due to the increase in the importance, timing, involvement, and scope of the testing, it becomes an area that can benefit significantly from the support of a BA. The chapter opens with a review of the agile approach to quality and it proceeds through a discussion of the expanded role of testing in agile projects. Additional areas this chapter covers include the all-important *definition of done*, the BA's support of the testing process, how to deal with defects, and challenges associated with agile testing. Due to the increased amount of testing on agile projects and the need for regression testing in every iteration, this chapter also looks into testing automation considerations and the benefits a BA can add in this area. The chapter concludes with a look into agile testing best practices and agile testing strategies.

Chapter 9 is one of the most unique chapters as it breaks down the day of the agile BA—item by item—as he or she supports the agile team. The daily breakdown does not attempt to dictate to the reader a prescription; rather, the chapter identifies typical activities and how they may all come together on a sample project. Your own project may require a different configuration; however, the sample provided illustrates the concepts to help you identify how to integrate agile business analysis practices on your own project. The chapter identifies three template days in the life of an agile BA: *the last day of an iteration*—where the focus is on wrapping up the iteration, demonstrating the product, getting feedback, incorporating feedback into the reprioritization process, and learning from what just happened; *the first day of a new iteration*—where the BA supports the team finalizing the planning and the setup of the new iteration based on the feedback and the results of the previous iteration; and *a typical middle day in the iteration*.

Chapter 10 cycles back to agile challenges and business analysis, but this time views them through a more discerning lens. The chapter discusses agile and social trends, reviews concepts around organizational agility, and looks into the future of agile and project management, the future of business analysis in agile projects, and the prospect of turning the BA into an agile center of excellence. The chapter concludes by reminding the readers that a BA must not act as a replacement for the product owner.

HOW TO BENEFIT FROM THIS BOOK

This book is an easy read and readers can focus on one chapter at a time. It has many practical examples and provides multiple practical ways to apply concepts, translate ideas and theories into benefits, and improvements to the way we manage agile projects. The content of this book reflects the authors' education and years of experience with applying the information, concepts, and ideas that are presented here—with proven results.

ACKNOWLEDGMENTS

Kevin Aguanno

My own journey to find more flexible and efficient ways of delivering high-change projects began in the 1980s. Early inspirations included Barry Boehm, James Martin, and Steve McConnell. Later, the writings of Kent Beck, Jeff Sutherland, and others provided additional insights. These (and others) led me to experiment with different approaches to defining and developing software-based systems and to combine these approaches with the project management techniques needed to ensure compliance with organizational governance processes while still allowing the project teams to be efficient in how they adapt to change. Despite my own pioneering work of combining agile estimating, planning, monitoring, and control techniques with traditional project management practices, I cannot take credit from where it is really due—I stand on the shoulders of giants in the field who set the groundwork for my own innovations.

I would also like to thank my co-author, Ori Schibi. It was his long hours and dedicated work that really made this book project happen.

Last, but not least, I would like to thank my wife, Alba, and my two children for their patience, support, and love.

Ori Schibi

To my co-author, Kevin Aguanno—a friend, a colleague, and an inspiration: it is a true pleasure knowing you and working with you and for you.

To my wonderful wife, Eva (who puts up with me), and our delightful daughters, Kayla and Maya: your support, the happiness you bring, and your love are more than what anyone could ask for.

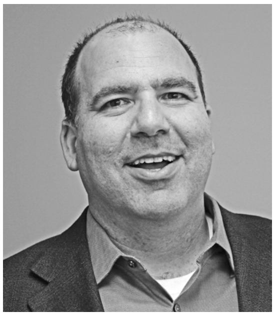
To my loving mom, Hava, and brothers, Eitan and Naaman.

In Loving Memory of Ruti Dalin (1939–2018), whom I have known my entire life: you were the closest thing to family there is. We will miss you and love you always.

ABOUT THE AUTHORS

Kevin Aguanno, PMP, PMI-ACP, CSM, has over 20 years of experience managing complex systems integration and software development projects. He is well known in the industry for his innovative approaches to solving common project management problems. He focuses on two project management specialty areas: agile project management and troubled project recovery.

As a well-known keynote speaker, author, trainer, and coach in agile management methods, Aguanno has taught thousands of people how to better manage high-change projects by using techniques from Scrum, Extreme Programming, Feature-Driven Development, and other agile methods. He is a frequent presenter at conferences and private corporate events where he delights audiences with practical advice peppered with fascinating stories from his own experiences in the trenches practicing agile project management. He is also founding Director and Vice President of the Project Management Association of Canada. He resides in Toronto, Canada.



Ori Schibi is President of PM Konnectors, an international consulting practice based in Toronto. With focus on project management, business analysis, and change management, his company provides a variety of services, including facilitation, workshops, training, and consulting—all of which deliver value to clients through a wide range of innovative business solutions. Large to mid-sized organizations in diverse industries and all levels of governments have benefited from PM Konnectors innovative services.

Schibi, PMI-PBA, PMP, PMI-ACP, SMC, is a thought-leader and subject matter expert in business analysis and both traditional and agile project management. He is a published author, speaker, and consultant with over 25 years of proven experience in driving operational and process improvements, software implementations, and complex programs to stabilize business, create growth and value, and lead sustainable change. He resides in Ontario, Canada.

At J. Ross Publishing we are committed to providing today's professional with practical, hands-on tools that enhance the learning experience and give readers an opportunity to apply what they have learned. That is why we offer free ancillary materials available for download on this book and all participating Web Added Value™ publications. These online resources may include interactive versions of material that appears in the book or supplemental templates, worksheets, models, plans, case studies, proposals, spreadsheets and assessment tools, among other things. Whenever you see the WAV™ symbol in any of our publications, it means bonus materials accompany the book and are available from the Web Added Value Download Resource Center at www.jrosspub.com.

Downloads for *Agile Business Analysis: Enabling Continuous Improvement of Requirements, Project Scope, and Agile Project Results* consist of:

- An Agile Triage Checklist
- An Example of an Iteration Burndown Spreadsheet
- An Agile Glossary
- PowerPoint Slides that Cover Key Agile Business Analysis Concepts
- Educational White Papers on:
    - Agile DevOps
    - Agile Methods and the Need for Speed
    - Agile and Capital Cost Appreciation

1

INTRODUCTION TO AGILE AND AGILE PROCESSES

Agile is not new, but its growth and increasing rate of adoption introduce new challenges (and opportunities) on an ongoing basis. Agile is also not the solution for all problems, but it is an approach, an umbrella, and a state of mind that, if done properly, can yield significant benefits for the performing organization. With that said, if agile is introduced the wrong way—in an unsuitable environment or under the wrong circumstances—it will cause more harm than good and there will always be someone who puts the blame on agile. While Chapter 2 of this book deals with challenges that emerge as part of adopting agile methods, this chapter introduces key agile concepts and examines the role of business analysis in delivering agile project success.

ABOUT AGILE AND WATERFALL

Agile is a pragmatic approach that recognizes that both the project team and the client do not know everything that has to be done up front, and that things will change along the way. Agile is an empirical approach—or, in other words, observation-based. It is open for changing needs and is evolutionary by nature, with multiple iterations (or sprints)—each producing a *shippable* product increment. The iterations are grouped into releases, where the customer actually receives, accepts, and, if needed, uses the released product increment. Agile addresses many of the challenges that were introduced by practices related to the waterfall approach, also known as a deterministic approach, where the team finalizes the requirements early on and then proceeds to build the agreed-upon product. Agile is made of the combination of two approaches: incremental (where product increments are released along the way) and iterative (where adaptation to change is continuous). Agile is not simply about breaking a large project into many small waterfall phases; there is more to it, thanks to the combination between the incremental and the iterative approaches.

It is not that agile is, plain and simple, better than waterfall. Although it addresses many bad practices that have been enabled by waterfall, there is still a place and time for a more deterministic approach. If a project has a set scope that is not about to change, and if the client does not need to benefit from early releases of the product by using it along the way, a waterfall approach may be suitable. Also, the fact that agile tries to fix some of the problems associated with waterfall does not mean that waterfall is at fault for project problems and is to blame for project failures. Waterfall simply enables some bad behaviors by allowing them to *hide* for too long before there are checks and balances to realize them. For example, because there are no interim releases and iterations along the way, the customer has the ability to see the product only at the very end—after testing has been completed—and only then can feedback be offered. It is possible that along the way the reports that provided the customer with updates were not portraying the actual picture, but the client had no chance to realize it because the reports were the only thing that the client had as *proof* of progress. This does not imply that the project team deliberately misled the client, but it is possible that information was communicated in a way that was misunderstood by the client. Good business analysis work and effective communication between the project team and the client can minimize the risk of such problems, but it provides no guarantee against such communication gaps.

Agile is an approach with several methodologies under its *umbrella*. Some of these methodologies include Scrum, Feature Driven Development, Extreme Programming (XP), Dynamic Systems Development Method (DSDM), Disciplined Agile Development, and Lean Development. While deterministic approaches focus on the plan and provide less flexibility, they also provide less chance of meeting schedule and cost constraints, which might lead to time and cost pressures that in turn may lead to quality problems. Empirical approaches, on the other hand, are evolutionary, observation based, and focus on delivering value by *locking* time, cost, and quality goals and focusing on scope flexibility.

In the early days of agile, around 2001, when agile was named *agile*, there was little to no talk about business analysis, or business analysts (BAs) in agile environments—and it was for good reasons:

- The role of the BA was still undefined—it was prior to when the International Institute of Business Analysis was founded (in 2003) and prior to when the Project Management Institute expanded its focus to include business analysis.

- The main premises of agile—including its focus on handling change, being nimble, reducing waste, improving communication, and building teams that are more cohesive—are all about reducing the need for a BA to *bridge* and connect different stakeholders, groups, and teams. It was therefore expected that agile role definitions include only three roles: (1) the product owner; (2) the Scrum Master, coach, or a role that is somewhat equivalent to that of a project manager (PM); and (3) a team member (meaning all technical members of the team). While some interpretations call for the inclusion of a BA as part of the team, this was not the original intent. Another interpretation of agile concepts agrees that there is a need for business analysis skills, but this need has to be addressed by team members demonstrating business analysis skills, without calling for a dedicated or distinct role of a BA on the project.

This book covers the growing and increasingly recognized need to have the skills of the BA present on agile projects. At times, we will discuss the need for business analysis skills to be demonstrated by a BA, and at other times, ensure that team members can demonstrate these skills. One way or another, it is clear that these skills are necessary and are important for project success, but there is no one clear and correct way to apply them. This means that for some project environments, it would be sufficient to have team members demonstrating business analysis skills, while other project environments need a BA to champion these skills and ensure they are applied consistently and correctly. In retrospect, as agile is closing in on completing its second decade, it is safe to say that although agile attempts to produce benefits that, if performed properly, lead to a reduction, if not total elimination of the need of BAs, agile growth and the way it is applied bring back those needs and with them—at times—the need for a specific role of a BA within the team.

One of the reasons that has rejuvenated the focus on business analysis skills in agile environments is the rapid growth of agile, its scalability beyond technology projects, and the attempts to introduce agile concepts in organizations and environments that are not a natural fit for agile—including slower moving financial institutions and government agencies. With more projects attempting to apply agile concepts and more team members assigned to those projects, there is a need to refocus the efforts and to add checks and balances that reduce the chance that these projects will fail. When more people within the organization practice agile and when more agile projects are competing for scarce resources, it leads to what we identify as *agile dilution*, where there is a decline in teams' relevant agile skills and a reduction in the true agile experts in the organization to oversee, support, and lead the agile process.

Before proceeding to discuss the basic concepts of agile, it is important to make the record clear: agile is *not* about the following things:

- Not about *working fast*: but rather, building products in such a way that we can produce value earlier.
- Not about *no planning*: agile planning is critical for success and it is simply done differently and throughout the project, instead of a heavy focus on building a plan early on. Agile focuses on the act of planning, rather than on the creation of a plan.
- Not about *no requirements*: similar to the planning, an important part of agile is requirements. While they are referred to by a different name—user stories—there is a need to elicit and manage requirements. It is done in a different fashion than what takes place in waterfall environments, but properly managing requirements is a critical part of agile success. It is the level of detail around requirements that changes since the project starts with high-level requirements and then the team elaborates and seeks details on a timely basis—like a rolling wave, only for the requirements that are up next in the cycle. The team and stakeholders need to come to terms with the fact that ambiguity is welcomed until more information is needed. In a way, the requirements are partially trial and error because the requirements that are defined at the start of the project may need to change later on as more information is discovered during the detailed planning. Agile is no different than waterfall when it comes to ambiguity. In both types of approaches, there is ambiguity and there is no full set of detailed information up front—it is just that agile teams recognize the ambiguity and accept it, while in waterfall, there is an attempt to finalize requirements and their details up front. This latter effort costs a lot of time, effort, and money, only to go through the likely need to change these requirements later on—in a process that not only costs a lot, but also one that adds a lot of risk. The reference made here to trial and error does not refer to the project objectives or the project vision, but rather to the process of building stories and refining their details.
- Not about *being careless*: many organizations attempt to apply agile concepts into projects the wrong way, in the wrong context, and for the wrong reasons—leading to uncertainty, confusion, and *wild-west* behaviors where team members careen forward irresponsibly, attempting to do things too fast, moving forward without planning, making reckless decisions, and causing more harm than good.

OVERVIEW OF AGILE CONCEPTS

Agile is a pragmatic approach that recognizes that there is a lot that the team and the customer do not know, a lot the team and customer learn along the way, and that things change without warning. Agile is a flexible approach that allows us to adapt and learn in real time as things happen, to respond to change, and to satisfy the customer as much as possible within the project constraints and realities. Agile projects start with a clear vision and product and are then broken into releases and iterations. The iterations should be of a consistent length (commonly, but not necessarily, between two and four weeks) and they are grouped together (any one or more iterations) into a release. Every iteration has a theme and goals and is made up of a slice of the project backlog (which is the total scope of the project—consisting of features, stories, and requirements). By the end of the iteration, the team needs to produce a working, production-quality product increment as if it was going to the customer. However, the product increment that is produced at the end of the iteration is for feedback purposes—the final product is actually delivered to the customer at the end of a release, with one or more iterations grouped together (and integrated through regression testing)—so this *chunk* of working product is a fully functional slice of the total product.

Known as iterative and incremental development, agile concepts help realize the following benefits:

- Adapt to changing needs, conditions, and environments in real time
- Produce benefits earlier through periodic releases of product increments
- Improve quality by giving the customer the chance to review progress periodically and improve based on actual performance and on the feedback provided—known as progressive requirement elaboration
- Reduce risk of building the wrong product and building the product with defects (the wrong way)
- Reduce scope efficiently through prioritization and reprioritization—agile enables the efficient reduction of scope by removing lower priority scope items in the later stages of the project once the majority of the value has already been delivered to the client

Additional characteristics of agile include accepting ambiguity in earlier stages until detail is needed, and ensuring that feedback is accepted as soon as possible so the team can properly act based on the feedback.

Lean Concepts, Wastes, and Principles

Additional agile concepts include the following:

- *Continuous improvement of processes (learning)—improving processes and practices on an ongoing basis*: This concept is inspired from the term Kaizen in Japanese (Zen = good; Kai = change). The continuous improvement refers to small, incremental improvements to be learned, realized, and applied on an ongoing basis, unlike concepts such as Kaikaku (from Japanese—radical change) which refer to dramatic and quick improvements. Continuous improvement takes place in small doses over time, which improves the ability to apply them one by one, reduces resistance, and helps isolate the impacts of the individual improvements.
- *Customer focus*: This is about producing and maximizing value for the customer.
- *Lean principles*: The lean movement started in Japan in the mid-1950s in manufacturing (the automotive industry) with its main focus on loss reduction and sustainable production. Figure 1.1 lists the *original* seven wastes identified in the Toyota Production System by Shigeo Shingo. Agile is one of the most significant and successful attempts to translate lean benefits from manufacturing to software development and, subsequently, to other service and non-manufacturing projects. Lean development can be summarized by the following concepts that are very close in concept to lean manufacturing principles:[1]
    1. *Eliminate waste*: Activities that do not directly add value to the finished product are waste. The three biggest sources of waste in software development are the addition of unrequired features, project churn, and crossing organizational boundaries. To reduce waste, the development teams need to self-organize in order to optimize themselves for the work they are trying to complete.
    2. *Build in quality*: There is a need to design quality into the product, instead of relying on later inspection. But when this is not possible, there is a need to break the work into small and more manageable cycles that involve work-validate-fix-iterate. Inspecting after the fact and queuing up defects is reactive and ineffective.
    3. *Create knowledge and amplify learning*: Promote strategies, such as iterative development, that help teams discover what stakeholders really need and act on knowledge as fast as possible. Leaning also involves frequent reflections and improvements.
    4. *Defer commitment* (*decide as late as possible*): This is not a call for procrastination, but there is no need to start building a product by defining a complete specification. The design and architecture

need to be flexible so they can change as needed. Deferring commitment to the last, most responsible moment allows the team and stakeholders to make the most informed decisions.

5. *Deliver quickly* (*deliver as fast as possible*): Quickly does not mean reckless; it is possible to deliver high-quality products quickly by identifying team capacity and limiting the work to this level. By becoming aware of its capacity (velocity), the team can establish a reliable and repeatable flow of work.
6. *Respect people and empower the team*: Sustainable advantage is gained from people who are engaged and thinking, and it is achieved by enabling and motivation, rather than controls and limitations.
7. *See and optimize the whole*: To achieve an effective solution, we must see the whole and the bigger picture, including understanding the high-level business processes that individual projects support across multiple systems and areas.

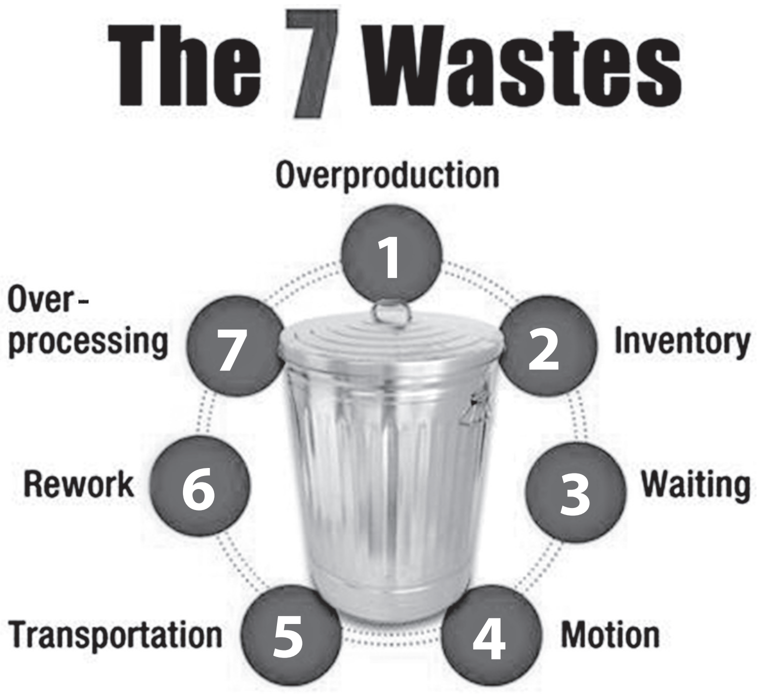


Figure 1.1 The seven lean wastes of software development: the seven wastes were identified in manufacturing in the Toyota Production System by Shigeo Shingo, then *translated* to software development by Mary and Tom Poppendieck

See Table 1.1 which illustrates the seven wastes of software development.

Table 1.1 Lean wastes: Shigeo Shingo (1981) identified seven wastes of manufacturing and Mary and Tom Poppendieck (2007) translated these into waste in software development. For an agile BA with keen eyes toward eliminating waste, this list offers a gold mine of opportunity

The Seven Wastes of Manufacturing	The Seven Wastes of Software Development
Overproduction	Partially done work
Inventory	Extra features
Extra processing	Relearning
Transportation	Handovers
Motion	Task switching
Waiting	Delays
Defects	Defects

Combating Waste

To better understand agile principles and concepts, it is important to review the ways lean attempts to fight waste. For that, we need to check the underlying reasons behind the waste and think about ways to reduce or eliminate that waste. Although this book refers to the benefits of a BA, or the application of business analysis skills in any type of project environment, the wastes we cover here are the translations by the Poppendiecks of the manufacturing wastes that are identified by lean principles to software development wastes. The following are the seven wastes, along with the potential causes and solutions.

1. Partially Done Work

This is work—including stories, features, and requirements—that is not done to the extent that qualifies them to be called *done*. Chapter 7 of this book discusses in detail what *done* means—but in short, *done* means that a feature, functionality, or work product does what it is supposed to do, to the extent that it needs to perform. Defining what constitutes as *done* in agile projects is critical for agile project success. Products and features that are not *done* cannot be released for the client to use.

Potential Causes for Partially Done Work:

- Issues with planning that compromise the team's ability to deal with technical issues, complexities, and dependencies
- Problems around prioritization and the information about the story

- Inconsistencies about following through with the iteration's mandate; removing and adding stories throughout iterations
- Lack of coordination within the team that leads to delays and bottlenecks
- Problems around task identification and matching of tasks to team members

Ways to Reduce or Eliminate This Type of Waste:

- Product owner involvement
- Proper prioritization and reprioritization for each iteration
- Ensure sufficient planning and estimating of stories and requirements (by the team members who actually perform the work on the respective stories); this will also help reduce *task creep*, where the actual work to complete a task ends up being more than the planned effort
- Follow through on the iteration plan and do not change stories (including adding and removing stories) midway through the iteration
- Since planning and prioritization includes managing story dependencies (including dependencies on external elements), ensure the product owner is involved in all stages of the prioritization and when the product owner lacks technical understanding, introduce a BA to support the effort and enhance the coordination and collaboration
- Cross-functional teams can help reduce roadblocks when other team members face problems

2. Extra Features

Extra features are also known as gold-plating, which is a type of scope creep—in short, it is about producing more than what we have been asked to do. Many people and even organizations confuse the term *going above and beyond* with *producing above and beyond*. Going above and beyond is about making a sincere effort to satisfy the client's needs. However, the intent is to satisfy the client within the scope of the work that was agreed upon, and it is not about producing more products or features. Gold-plating is about giving the client more than what they signed up for. As an example, let's think of a chair manufacturer that was requested to deliver a certain number of office chairs. The contract specified the number of chairs, models, trims, finishes, colors (all under the umbrella of scope), costs, and timelines. When the chair maker delivered the chairs, the customer was surprised to see that the chairs that were delivered included the same features of a higher-end and a more expensive model than what was requested. This means that the chairs had more features and were made of a more expensive (and apparently more refined) upholstery. The chairs arrived on time and there was no extra cost to the customer.

Should the customer be happy? The answer is no. The customer asked for the specific chairs for a reason, since the chairs that were requested addressed the customer's needs. Any additional features were not required, asked for, or welcomed. Because the extra features were not in the contract, it may be that they were not designed or installed properly, which will potentially deem the chairs as unusable. It may also mean that the extra features are not appropriate or useful in the customer's environment. For example, it is possible that these are additional chairs to an existing office, and by bringing in new chairs, it may cause conflict in the office as to who will receive the new, improved chairs. Although the customer was not asked to pay more for the chairs, it does not mean that the customer views these chairs as being of a higher value. Further, the delivery of the higher-end chairs may set a precedent in the eyes of the customer that in the future there will be an upgrade every time. Finally, if the customer continues to deliver high-end chairs for lower prices, they may incur losses and may go out of business.

Any way we look at it, there is no benefit in delivering more than what the customer asked for. If the chair maker wanted to do it the right way, they should have asked the customer whether they were interested in the special deal of the higher-end chairs for a lower price. Although this example is simplified, gold-plating happens often and in many cases is not realized until it is too late. Gold-plating can occur on the team level, where team members may decide of their own volition to add features, thinking it might be a good idea to build something in addition to what was required, or to apply their understanding or logic to interpret customer needs. It is clear that it is all in the name of delivering customer satisfaction; but if there is a situation where more features or more things can be produced for the project's work, it is important to communicate and handle it properly by ensuring that there is customer acceptance and understanding of all the related implications.

Team members' awareness of business analysis skills can help in these situations so the team members will perform the work as agreed upon and use the proper channels for any potential additional features. Alternately, a BA is simply another set of eyes to ensure that the work performed is in line with what was required and requested, and that any opportunity for additional features is communicated, addressed, and goes through the proper channels for due diligence, approval, prioritization, and planning.

Potential Causes for Extra Features:

- Lack of clarity around the product vision
- Issues with prioritization of features
- Gold-plating
- Lack of understanding of the true customer needs or product use
- Lack of discipline among team members

Ways to Reduce or Eliminate This Type of Waste:

- Assure product owner involvement throughout—provide clarity around the product vision, reiterating the vision and any adjustments on an iteration by iteration basis
- Communicate the nature of the value to be produced and create awareness around the return on investment of features within the project
- Provide clear prioritization and reprioritization of stories and features; Chapter 7 of this book presents a detailed review of the stories' and features' prioritization process, but in short, prioritization has to be consistent and is based on the following criteria:
  a. Business value (provided by the product owner)
  b. Technical considerations and dependencies (identified by the team and addressed through discussions between the team and the product owner)
  c. Logical groupings (identified mainly by the team; may include considerations related to product viability)

Either way, the product owner has the final say about the prioritization of the backlog items.

- Since gold-plating is often originated within the team and for the most part is unintended, prioritization, open communication, and clarification of the value of features is important—teams can ensure these items are properly addressed with the help of a BA, or the application of business analysis skills

3. Relearning

This is the failure to use knowledge that is available to improve the way the team works or to refrain from stumbling into known (or previously encountered) issues and problems. Relearning also involves team members who attempt to solve problems (reinvent the wheel) on their own, instead of utilizing a solution that is already in place.

Potential Causes for Extra Relearning:

- No proper knowledge sharing within the team
- No proper lessons-learned process at the end of iterations
- Teams that are not colocated
- Issues surrounding communication, including agile reporting
- Problems with (and specifically missing) documentation

Ways to Reduce or Eliminate This Type of Waste:

- Colocation is not always possible, but it is proven that colocated teams perform better than distributed teams. When there is no choice, there are additional factors that need to be introduced to make up for the distance: strong leadership, focus on communication for distributed teams, application of sharing and reporting tools, establishment of expectations, and ground rules for the team and other stakeholders. Did we say leadership? This consists of a product owner that is present, a strong Scrum Master, and most likely the need for a seasoned BA since the sharing of the task to apply business analysis skills among team members may not be successful. This type of sharing is challenging with teams that are colocated, and it becomes significantly more challenging in distributed teams.
- Knowledge sharing throughout and at the end of the iteration is important.
- Properly run daily stand-up meetings will be beneficial.
- Provide real time and continuous updates to information radiators, task boards, and other reporting tools.
- Require participation in team meetings and buy-in to the agile processes and ceremonies.
- Develop documentation. Agile is not about having zero documentation: Chapter 5 covers agile documentation in detail, but we will go back to the idea that documentation is and has to be part of an agile project—multiple times. Documentation is the responsibility of the entire team and it has to take place on an ongoing basis, just-in-time.
- Identify areas (physical and virtual) that are accessible for the team to view, share, and discuss ideas, findings, and performance.
- Perform meaningful retrospectives where team members participate and contribute to the effort. Findings should be established by consensus, applied soon after, and reviewed by the end of the next iteration. The BA can facilitate the retrospective meeting, provide feedback and input to discussion items, monitor the application of improvement ideas, share the results, and report on related issues.

4. Handoffs

A handoff is the handing off of work from one person to another. Lean calls for the minimization of handoffs because they create inefficiencies and open the door for mistakes and misunderstandings. This is not a call to have only one person doing the work, but the intent is to reduce the number of unnecessary handoffs to the minimum necessary.

Potential Causes for Handoffs:

- Some tasks and activities require handoffs
- Lack of availability of, and access to, information
- Distributed team members working on the same items
- Planning problems: issues that arise as a result of poor planning—the need to move tasks from one team member to another, escalations, misunderstandings, and lack of team rhythm

Ways to Reduce or Eliminate This Type of Waste:

- Certain tasks have to be performed by multiple team members; each one brings in his or her own unique experience and role.
- With good planning, the need for a handoff, even when there is a need for cross-functional teams, can be minimized. Proper identification of needed cross-functional skills and resources can also help reduce handoffs.
- Distributed teams typically have challenges with handoffs, especially when there are time zone differences. These handoffs lead to a different type of waste—delays. Focus on communication; coach the team to ensure that all team members own their part of the communication; establish practices for proper sending, receiving, and feedback cycles for messages; and monitor how it takes place for potential improvements.
- Try to perform related tasks and work on specific features in one location.
- Proper updates of task boards, reporting, version controls, and availability of tools, such as wireframes and flowcharts, can also help reduce the number of handoffs and related challenges.

5. Task Switching

Task switching is often mentioned in the context of multitasking. Many people confuse the terms and pride themselves for being able to multitask. While it is important to be able to handle multiple priorities simultaneously, this is not the meaning of multitasking. The large majority of people cannot multitask. Multitasking is about doing two things at the same time with the same part of the brain. While (most) people can walk and chew gum at the same time, when it comes to activities that compete over the same part of the brain, it is a different story. Almost any project practitioner, or even virtually any person who has an office job, has gone through the attempt to do multiple things at the same time. The most common example is to dial into a conference call, then muting the line and trying to do work at the same time. It is so common that we often do not give it a second thought, but the question that needs to be asked is whether or not it benefits us. When we do this type of multitasking, we do not get the level of focus that is needed to perform the tasks we have in mind, and at the same time, we do not pay sufficient attention to the call. The result often is that the work does not get done; someone on the conference call asks us a question, but we do not hear it; the second time they call our name, we respond, but with the phone still on mute; and on the third attempt, we finally unmute the phone and ask them to repeat the question. By now, our train of thought is cut, and as for the conference call—we do not know what is going on there, and the other participants are most likely irritated at us by now for the delay and distraction we are causing.

This was just one example, but it has been proven both scientifically and practically that multitasking not only does not help productivity, but rather hurts it. A study by the American Psychological Association[2] found that multitasking reduces task efficiency and is especially hard on our brain power when we switch to difficult or unfamiliar tasks. It is estimated that we lose up to 40% of our productivity when we multitask, and every interruption *costs* us an additional period that ranges between five and 25 minutes—in addition to the time it takes us to attend to the interruption. To illustrate, if someone is focused on writing a report and an e-mail pops up on the screen, once this person attends to the e-mail, it will take him/her up to 25 minutes after finishing with the e-mail to get back to the level of focus and rigor he or she had prior to the interruption. When adding it all up, how many interruptions can we handle in a day before our productivity becomes a write-off?

When working on an important task, it makes sense to have a second task to work on in case we hit a roadblock with the *original* task. It can help improve our productivity to have a second task lined up to work on, so we do not sit idle while we wait for the first task to resume. However, from here, the law of diminishing returns kicks in rapidly—once we add additional tasks, our productivity plunges significantly. When we have multiple items on our plate (and we all have multiple things to do), it is important to identify our own capacity so we know our limits and what we can do within those limits. Once our work volume reaches the limit, it is time to prioritize our work and determine the sequence by which we will perform the work. We should then focus on one thing at a time based on the priority that is set and proceed with it before moving on to the next task.

Discussing personal productivity, capacity management, and prioritization may not appear to be directly related to agile projects, but these items are in fact key success factors in agile projects. Teams must determine their capacity (measured by velocity), stories and features must be prioritized, work must be capped at the capacity, and team productivity must be maintained at the expected pace sustainably, over time. The BA, along with the Scrum Master, should be both role models in productivity and coaches in applying these concepts (productivity, capacity management, and prioritization) for team members. At the same time, the BA and the Scrum Master should also facilitate the prioritization process and the process of determining and maintaining velocity.

Task-switching becomes an even bigger problem when team members move from one task to another without completing tasks properly during the process, leaving a lot of work in progress (WIP).

Potential Causes for Task Switching:

- Interruptions are a major cause that leads to task-switching. Interruptions are the result of a variety of causes including poor planning, changing priorities within the iteration, a mismatch between resources and tasks, and performance issues.
- Lack of coordination between the product owner and the team. This could be driven by a product owner who is not sufficiently involved or due to communication and reporting issues
- Shared resources when the team works on more than one project. An important success factor for agile projects is having a dedicated project team. While it is possible to share resources with other projects, it often leads to problems and delays since the project would then be exposed to problems, both from within as well as to any issues that hold the resources in the other project.
- Planning issues, such as when task estimates for the iteration are not done properly and the team struggles to meet its commitments.

Ways to Reduce or Eliminate This Type of Waste:

- Make the team dedicated.
- Ensure proper planning, estimating, and sufficient breakdown of the stories into tasks. Also, it is important to ensure that team members match themselves properly with tasks. Sequencing of the tasks is also important in order to allow stories to finish without bottlenecks and with a minimal number of task-switching incidents.
- Reduce the number of interruptions by ensuring that all information about tasks and priorities is clearly communicated and understood, and that tasks are estimated and detailed sufficiently. In addition, identify in advance subject matter experts (SMEs) to be accessed for each story, as well as external dependencies. The BA and the Scrum Master need to facilitate these areas and work with the team to minimize these areas of impact.

6. Delays

Delays slow down the team's value creation and lead to frustration-causing further distractions and interruptions to the workflow. Delays are events, problems, misunderstandings, and slowdowns that cause more time to deliver or to start the work on value-added activities.

Potential Causes for Delays:

- Process issues—when there are bottlenecks and misunderstandings in processes.
- WIP—when there are too many things on the go, or in other words, too much work in progress. This can be driven by performance issues, by unrealistic expectations, or by poor planning.
- Resource sharing and other external dependencies—the team cannot move forward with its tasks due to lack of resources or an extensive need to coordinate the work with external stakeholders and team members.
- Uncertainty—meaning when there is a high number of items that require clarifications, when many impediments and hurdles introduce themselves, and when there are many mismanaged assumptions or assumptions that are not validated.

Ways to Reduce or Eliminate This Type of Waste:

- Efficiencies—identify only the processes that are necessary for the team to produce value for each iteration. Anything that can help reduce the cycle time of tasks and activities should be considered.
- Identify external dependencies and support needs in advance (for each iteration)—it is important that SMEs, stakeholders, and other support functions from outside the team are readily available in a timely manner. This requires careful planning and the ability to identify needs at almost exact timing. The need for proactiveness and coordination is further reinforced with the distributed team; and it should be supported by accessing the supporting resources in advance, if possible, and by always having a backup plan and contact information in case there are connectivity and communication issues.
- Prioritization—beyond story prioritization, it is important for team members to properly prioritize their tasks. Effective prioritization will help improve the overall planning process and allow other team members to pick up work that is based on their skill levels and capacity, helping their colleague where and when there is a need for such help.
- Resource availability—ensure that all required skill sets are available and assigned to the project when required.

- Check processes, performance, and efficiencies—a value stream mapping exercise can help in determining what is value-added time and what is not in order to help improve cycle time.
- Automation—automate test cases and deployment where possible (while considering the cost-benefit) to help reduce cycle times significantly. This is not a call to automate all test cases, but rather to automate those that are recurring and can be reused. Chapter 8 of this book elaborates on the area of testing and provides important information regarding testing automation considerations.
- Manage assumptions—when asked, most people say that they do not like assumptions. That is despite the fact that we all make assumptions. Furthermore, we heavily rely on assumptions when we plan. Managing assumptions means that when we make one, we track it in an effort to validate it. When an assumption is not validated (whether there is no answer or the assumption does not end up playing out the way we assumed), it becomes a risk and should be managed systematically, the same way we manage other risks.

Table 1.2 illustrates how to manage assumptions by providing three columns: (1) what we need to know; (2) who needs to provide us with information; and (3) when we need to know it. Managing assumptions (typically the Scrum Master with support from the BA) can make a significant contribution to the team's ability to handle situations, and proactively identify risks and address issues.

Table 1.2 Project assumptions: three columns that if identified, tracked, and managed properly, can dramatically improve the project team's handle on issues and risks, and help the team become more proactive

Assumption: *WHAT* we need to know	***WHO* needs to provide the information**	***BY WHEN* do we need to know it; if the answer is unfavorable, or there is no answer by this deadline, it becomes a risk**

7. Defects

Defects (also known as bugs) are errors in a product's functionality that lead to the production of the wrong result. Defects are the epitome of waste since they are produced despite all the efforts to design, build, and test a product. The waste consists of the need to undo and redo the work, increasing costs, and inevitable delays which compromise the value that the team produces, thereby reducing customer satisfaction. All these considerations come in addition to the direct impact of the defect, which may cause multiple problems and issues.

Potential Causes for Defects:

- The story is not a good story—stories need to be independent, negotiable, valuable, estimable, small, and testable (INVEST) in order to be considered as *good stories*
- Stories have no acceptance criteria
- Lack of sufficient technical standards
- Issues around understanding stories
- Insufficient available skills among team members
- Insufficient or late involvement by testers
- Not enough testing automation

Ways to Reduce This Type of Waste:

- Establish a clear definition of *done*
- Follow the INVEST principle
- Break stories down sufficiently
- Clarify the acceptance criteria for each story with the product owner
- Prioritization to ensure that the team's effort works toward producing value
- Ensure the right skills are available on the team (and in a timely manner)
- Establish relevant and sufficient coding standards and guidelines
- Ensure clear understanding of stories prior to the start of work on these stories
- Follow agile principles—in this case, simplicity, in order to maximize the work undone
- Follow agile practices that are suitable for the project's needs, the organization, and the team
- Involve the testers from the start—ensure the testers are part of defining acceptance criteria for each story and that the test cases are written based on the acceptance criteria
- Testing automation

Waste Summary

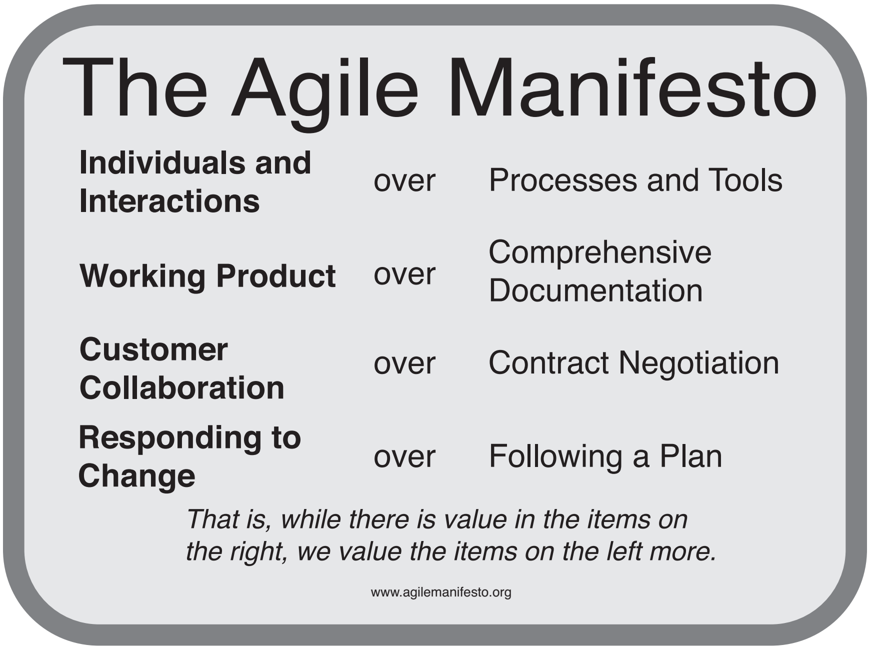
The wastes, causes, and proposed high-level solutions discussed here are part of our introduction to agile concepts. Chapter 2 of this book takes a deep dive into agile challenges and into ideas on how to address these challenges. The items discussed in Chapter 2 are directly related to agile projects, while the items discussed in this section are addressing only the lean wastes. Although aligned, they are not all the same.

Back to Agile

In February 2001, members from XP, Scrum, DSDM, and other methods got together in a ski resort in Utah—including luminaries Kent Beck, Ken Schwaber, Alistair Cockburn, Jim Highsmith, and Steve Mellor—and decided to each keep their respective methodologies and their characteristics, but put them all under the same umbrella and call it *Agile*. They also produced a set of value statements about what their methodologies stood for and they named it the *Agile Manifesto*, shown in Figure 1.2. It is important to read the Manifesto as it is intended to be and not out of context; keeping in mind the very important statement at the bottom: *while there is value in the items on the right, we value the items on the left more*. The items on the right are foundational, and critical for success, but the items on the left are the ones that will take us to the next level. There are many ways the BA can help in delivering on the value statements expressed in the Manifesto, so let's take a quick look at the benefits the BA can add toward achieving those values by helping to attain the items on the left, but also ensuring that the items on the right are in place:

- *Individuals and interactions over processes and tools*: Processes and tools need to be in place to provide some predictability and familiarity with how to conduct the work and what to expect. The BA can help provide the team with access to processes, information, and context; help improve and streamline the process; as well as facilitate adjustments in application of the processes in the agile project environment.
- *Working software over comprehensive documentation*: While the original intent of agile was to improve software development, agile has grown to be applied in essentially any type of project, so we will keep referring to concepts in the context of any type of product, and not only software. With that said, it is not easy to produce a working product and there still has to be sufficient documentation and development of prototypes, wireframes, or proof of concepts. The BA can help facilitate these elements and provide support for the team to maximize the value they produce.
- *Customer collaboration over contract negotiation*: It is hard to achieve true and meaningful customer collaboration, but it is important to note that the need for customer collaboration is important in any type of project life cycle—agile or not. Sticking to the word of the contract rarely yields the desired results without working closely and collaboratively with the customer in order to truly understand their needs. BAs were originally introduced to improve customer collaboration; their role is to ensure that it takes place and to facilitate the process by providing the team with the necessary support and enablement.

- *Responding to change over following a plan*: This is one of the most important aspects of agile—the ability to respond to change effectively, in real time, and at any time throughout the project. Business analysis skills are important to ensure change accountability, so when the scope changes, timelines need to change too. With the concepts of time-boxing, an iteration (and potentially, ultimately, the project) should not get extended, but the team needs to have the ability to account for changes and to identify the impact of these changes. This takes us back to the need for capacity management (measured by velocity) and prioritization. Ideally, team members should be able to account for the impact of changes, but depending on the project context and the specific business analysis skills of the team members, a BA is often required to help ensure that all scope changes are accounted for.



The Agile Manifesto

Individuals and Interactions	over	Processes and Tools
Working Product	over	Comprehensive Documentation
Customer Collaboration	over	Contract Negotiation
Responding to Change	over	Following a Plan

That is, while there is value in the items on the right, we value the items on the left more.

www.agilemanifesto.org

Figure 1.2 The Agile Manifesto: keep in mind the very important statement at the bottom

Agile Principles

Agile project characteristics include early and continuous delivery of usable deliverables through short delivery cycles and simplicity. Agile recognizes that usable deliverables are the best measure of progress and of value creation for the customer—and it accepts that requirements change, even late in the project. In agile projects, business people are involved daily with the project team to maximize the use of face-to-face conversations, which sometimes introduces challenges. While there is no reliance (like in a waterfall project) on the BA to ensure such communication between the business and the team takes place, a BA may need to facilitate certain elements of these interactions.

Team members in agile projects are motivated, trusted, experienced, and self-organizing. While that is a good thing, many people tend to misinterpret it by envisioning projects with no BAs. There is obviously nothing wrong with having teams that are *qualified* according to all of the descriptions mentioned, but in reality, many teams fall short of these qualifications, and this is where BAs can and need to become part of the project. In fact, for the team to be self-organizing, most team members need to possess and demonstrate business analysis skills. Ideally, agile projects should not need specific individuals whose titles are BAs, but the circumstances surrounding many projects dictate that need. Our goal is not to force the use of BAs in projects, but rather to recognize when a BA is needed and to what extent. The opposite also applies: while agile projects are more likely to get by and succeed without a dedicated BA (than waterfall projects would), we should not force the removal of BAs from agile projects just for the sake of having no BAs in and of itself. BAs can also facilitate the team's learning process through the application of lessons learned and the ongoing focus on continuous improvement.

Tools and Techniques

Let's review a few high-level tools and techniques that are relevant for agile projects. These will be discussed and elaborated on in detail throughout the book:

- *Agile planning*: Agile values planning and the act of planning more than the plan that is being produced. It is important to differentiate between the two aspects. While waterfall is big on the plan that is produced at the beginning of the project, agile is about value-creation and planning on a high level up front—with details being figured out just in time. Planning, monitoring, and adapting to change are about creating manageable and lower risk cycles (following the Deming Cycle of Plan-Do-Check-Act and focusing on the concept of rolling wave planning where we plan a little, do a little, learn, and adjust).
- *Agile estimation*: Like planning, estimating is also about the activity itself, which is a team-building activity. There are estimating techniques that are engaging and they are performed in such a way that it increases the chance of producing realistic estimates. There are also multiple checks

and balances around estimating (and planning) in order to move from high levels to lower levels and to refine our understanding of the needs. Chapter 6 of this book focuses on agile estimating.

- *Ambiguity until further information is required*: We need to plan only on a high level and provide details only for the short-term planning horizon. Simplicity and value-creation are key, but it is not about procrastination—deciding later means that by then, we will have a better understanding of the solution (the big picture) and how the requirement fits in, there will be fewer unknowns as uncertainty is resolved over time, and we will end up spending less time on understanding requirements that will change or be canceled anyway. However, we must keep in mind that we still need at least a high-level set of requirements to help prepare an overall solution outline/model.
- *Communication*: Agile relies heavily on clear and effective communication that is delivered directly, timely, and with transparency. It includes reporting mechanisms (in real time, called information radiators) to efficiently and clearly convey to all stakeholders the most recent and accurate set of information.
- *Interpersonal skills*: With the transparency and the ongoing involvement of the customer, there is a need for strong interpersonal skills and the ability to manage ambiguity, stress, uncertainty, and conflicting priorities. While the role of the Scrum Master (or PM) is to protect the team from external interruptions and to lead the communication process with stakeholders, the BA can help create a bridge between the customer and the project team, as well as within the team and with internal functions and SMEs.
- *Continuous process improvement*: The team should learn from lessons in real time—every iteration. We do not have the luxury to wait until the end of the project in order to apply improvement initiatives.
- *Product quality*: Every iteration produces a production-quality product increment that is ready for shipment to the client. It does not mean that the client will want to get a product increment at the end of each iteration, but whatever we do must minimize WIP and be as good as if it is going to be delivered to the client. Iterations are grouped into releases of meaningful product increments that represent a critical mass of value for the client that justifies the effort to deploy and use the product in the interim.
- *Time-boxing*: Iteration (or Scrum sprints) are time-boxed, which means that they have a preset duration. This duration should reflect the project and customer needs, the nature of the product, and the team's velocity

(its rate of producing value). Once we understand our velocity, we can determine how many features will fit into an iteration, and by multiplying the number of iterations, we can provide a fairly accurate time frame for finishing the project.

- *Risk management*: Agile focuses on reducing risk—the risk of building the wrong product and the risk of building the product the wrong way (with defects). Also, thanks to checks and balances, there is a better ability to identify early on how we progress; and if the progress is not satisfactory, it is better to realize it early on rather than later.
- *Value-based prioritization*: Prioritization of stories (as well as requirements and features) is critical for project success. It is done and determined by the product owner and the most important factor in the prioritization process is whether an item represents business value.
- *Burndown charts*: Progress is measured through burndown (or burnup) charts. These are graphs that show the plan versus actual velocity, pace of value creation, and completion of stories. It is a clear indication of the rate by which we progress toward finishing the project and it provides an opportunity to act proactively and in real time on changes in the velocity or the needs.

THE BA, EFFICIENCIES, AND AN EYE FOR WASTE

Business analysis skills are a key to project success; no matter what type of project or life cycle we manage it by. The premise of agile principles is that team members need to have and demonstrate business analysis skills, including the following elements:

- *The ability to pick up stories*: As part of being self-directed, agile calls for team members to assign themselves (or volunteer) to work on stories (requirements) that are in the backlog. It should not be part of the role of the Scrum Master, or coach, to assign work to team members since there will be more context and buy-in if team members pick up stories to work on based on a combination of the project's needs, priorities, story complexity, experience, relevance, the applicable skills and experience of team members, and other things that need to be done on the project. Having team members consider and act on what is the best story to work on requires them to have a lot of context and understanding of the big picture. Having people picking up work on their own provides an additional benefit—enhanced buy-in—as it is likely that team members pick up items that they desire and are qualified to perform.

While it is not implied that technical team members lack such skills, it is important to keep in mind that in many environments, team members will not have the time or capacity to perform such due-diligence analysis—that essentially optimizes the story selection to all other moving parts around us. For situations where team members cannot pick stories on their own, there is a need for a Scrum Master/PM that will help facilitate this process; and this is typically done through collaboration with the BA, who often has additional insights, context, resource availability information, prioritization consideration, and other relevant decision support information.

When the "fathers" of Agile got together in 2001 and named the three agile project roles (i.e., product owner, Scrum Master/coach, team member), they were adamant about not including the PM as one of these roles. It was important for them to differentiate agile projects from *traditional* ones by excluding the PM from it so that they would not imply that the role of the PM is about bossing people around, putting out fires, policing team members, or *herding cats*. With a set of technically excellent, self-directed team members, it was an effective way to differentiate the characteristics of an agile project from that of waterfall projects: the role of the Scrum Master/coach is to add value by protecting the team and being a servant-leader, rather than being there as a safety net for a team that does not perform to the extent that it needs to.

While the intent behind the *absence* of the PM role from the three agile roles is clear, the same cannot be said about the role of the BA. There is a strong need for business analysis skills, and if these skills are not available across the team, there is a need for a BA to complement the situation.

- *Coordination and collaboration*: We need to consider these items in more than one context—coordination with colleagues, within the team, across the organization, with SMEs, with stakeholders, externally (with vendors), and with the customer/product owner. It involves clear prioritization, the escalation process, clarifications, estimating, the impact on other areas, story selection, task identification, changes to performance, and changes in needs. There is a need to ensure seamless collaboration between the developers and the testers, between the developers/testers and designers/architects, and between those who identify requirements and the rest of the team. With fast-tracking being common (doing requirements and design work one iteration in advance for the next iteration), it is often not sufficient to leave all the coordination and collaboration in the hands of the team members, and it may require a BA to facilitate some of these efforts. There are a lot of moving parts in agile projects, and even though there are not supposed to be changes to the scope midway through an iteration, team members may end up being too focused on their tasks, rather than on the handoffs, coordination, and collaboration.

- *Communication and reporting*: Agile projects introduce a different way to communicate and report on project performance. There is a lot more transparency, real-time information that flows, and direct contact between stakeholders and the team. Naturally, these conditions may take some team members out of their comfort zone since they may not be comfortable with the real-time visibility into what they work on at any given time, with the customer's involvement, with the constant need to provide updates, or with the less formal ways of reporting and communicating. For many, formality and documentation provide a sense of security and comfort, and the agile project typically takes that comfort away from many team members and stakeholders. When people do not feel comfortable with the processes around them, they may either take shortcuts or default to old ways of doing things. The impact on other people may be in the form of stress and performance issues. Most of these issues present themselves when team members lack business analysis skills or when people run out of time or capacity to demonstrate their business analysis skills.

  In uncertain situations, people tend to focus on their primary role and push aside secondary chores they need to perform; and with business analysis skills falling under the latter category, those will be the first things to go. Here, too, the intent behind having a BA within the team is not to pick up slack that people neglect to perform or that is a result of carelessness, but rather to complement team members' skills and provide them the necessary support at their weak points so they can focus on their core role and maximize the value they produce.
- *Eye for waste and process improvement*: One of the core traditional roles of the BA is to keep an eye on waste and to identify and implement ways to improve processes. Earlier in this chapter we discussed the common types of wastes identified by lean and we saw how the BA can help alleviate these situations, especially when things change in the organization and when teams transition into agile projects. Team members may not have the capacity or priority to focus on efficiencies, waste reduction, and continuous improvement. While these things are important, they are not always tangible, and they are not part of the core role of any team member. As such, similar to the other items discussed in this section, team members will scramble to produce the work products associated with their core role. They will then focus on delivering tangible

deliverables (or work products) and only then (typically the last item on the list) will they get to the efficiencies and improvements. With these items being part of the core roles of a BA, we should expect the BA to virtually always pay attention to these items. Another argument for the need of a BA is that leaving the improvements activities in the hands of many may end up where no one pays true attention to them, and as a result, improvements may never take place—at least not until they become necessities, pains, or problems.

Organizations must check whether there is sufficient capacity for team members to perform business analysis activities—or identify someone to do that for the team. Just by labeling a project as agile, we do not guarantee that everyone will do exactly what is expected of them, especially when it does not go beyond their core traditional role. Part of the continuous improvement activities of the team is achieved through the retrospective where it is important to capture, document, and apply lessons learned, as well as to track and report on the extent of the improvements. When the team does not have the capacity to do so, a BA needs to facilitate these activities.

- *Adapt to change*: One of agile's core intents is to handle change and this is done through communication and collaboration with the product owner and the activities around backlog maintenance. A few situations may introduce the need for a BA: (1) when the product owner is not sufficiently available, a BA can act, partially, on behalf of the product owner (though not for determining business value); and (2) someone (or more than one person) within the team needs to capture the feedback from stakeholders that is received during the demo and translate it into actions and updates of the project backlog (this is backlog maintenance). Further, once there is approval for the changes on the backlog, team members need to move forward with the planning, task identification, estimating, and coordination process. When team members do not have the capacity to go through these activities, once again—a BA can help.

RECAP: AGILE CONCEPTS

Agile methodologies are designed to overcome many of the challenges that *traditional* project life cycles enable (e.g., waterfall or predictive life cycles). Although there is no clear mention of the need of a BA in agile projects, in many environments having a BA as part of the team may be one of the keys for success.

First, by trying to overcome problems and do things differently, agile projects change the way teams do their work, introduce efficiencies, and require team members to demonstrate business analysis skills. Where team members lack these skills or where processes do not enable the demonstration of these skills by team members, there will be a need for a specific BA role.

In addition, like any other type of improvement, agile projects introduce new challenges that if not addressed may become issues and problems. These new challenges are triggered as a result of doing things differently and they may appear as *side effects* of the team's performance. They can range through pretty much anything—from estimating challenges, coordination, the ability to see the whole picture, ambiguity, different (and typically reduced) documentation, or changes in reporting and communication. Whatever it is, a BA can help in closing the gaps and cracks these issues may create while complementing the skills and experience of the team members in the process.

Whether the intent is to utilize the business analysis skills among team members or to introduce a BA role to be part of the project team, the intent of agile business analysis is for a more enhanced business analysis role that complements relevant skills of the team—as opposed to some of the older views of the BA as a bridge and a translator to make up for skills deficiencies, communication breakdowns, or lack of attention to detail by team members. While ideally the agile team should be comprised of technically excellent, motivated, empowered, and self-directed team members, in reality, the setup is not always sufficiently conducive to achieve the project's goals and at times it needs to be enhanced by ensuring business analysis skills are demonstrated—one way or another.

There is no one magic way of looking at things to determine whether there is a need for a BA in a project or that the team has what it takes to cover the business analysis activities and areas. Similarly, some of these skills may be filled, fulfilled, or supported by the Scrum Master, coach, or PM. However, we must make sure that the person leading this project has the knowledge, capacity, and context to do so—without compromising other aspects of their role.

ENDNOTES

1. http://www.disciplinedagiledelivery.com/lean-principles/.
2. http://www.apa.org/monitor/oct01/multitask.aspx.