CHARLES G. COBB

THE PROJECT MANAGER'S GUIDE TO MASTERING AGILE

PRINCIPLES AND PRACTICES FOR AN ADAPTIVE APPROACH

WILEY

›The Project MANAGER'S GUIDE TO MASTERING AGILE

The Project MANAGER'S GUIDE TO MASTERING AGILE

Principles and Practices for an Adaptive Approach

Charles G. Cobb

WILEY

Cover image: Blur © iStock.com/snvv
Cover design: Wiley

This book is printed on acid-free paper.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada

ISBN: 978-1-118-99104-6 (paperback)–ISBN 978-1-118-99177-0 (epdf)–ISBN 978-1-118-99176-3 (epub)

Printed in the United States of America

CONTENTS

PREFACE

THE PROJECT MANAGEMENT PROFESSION is beginning to go through rapid and profound changes due to the widespread adoption of agile methodologies. Those changes are likely to dramatically change the role of project managers in many environments as we have known them and raise the bar for the entire project management profession.

It is not a simple matter of making a binary choice between a totally plan-driven approach and totally adaptive or agile approach. There are many alternatives between those extremes, and it takes a lot of skill to adapt an approach to fit the situation. This book is designed to help project managers with a traditional, plan-driven project management background understand these challenges and to develop a more adaptive project management approach for blending traditional project management with agile principles and practices in the right proportions to fit a given project and business environment.

Agile is changing the way we think and work in many industries and application areas. The impact today is most obvious in the area of software and information technology, where an agile approach is essential to deal with the level of uncertainty in a typical software development project; however, the rapidly changing and competitive business world we live in today is already beginning to rapidly expand the influence of agile to many other areas.

This is the third book I've written on the subject of agile project management. My primary motivation in all of the books I've published in this area has been to help close the gap between the traditional project management and agile communities. Those two areas have essentially been treated as separate and independent domains of knowledge with a very limited amount of integration between the two and some new thinking is badly needed to see both of these areas as complementary to each other rather than competitive.

If I were to publish this book as an entirely separate and independent book from my two previous books, it would have either been disjoint or there might have been redundancy with the material in the two previous books. For that reason, I have decided to merge together some information from my two previous books into this one book to make it much more comprehensive, well-integrated, and easy to follow. It is designed to be used as a textbook in a graduate-level Agile Project Management course and includes a suggested course outline and instructional materials to align with the material in the book.

THE IMPACT OF AGILE

I believe that agile is having a profound impact on the project management profession and will cause us to fundamentally rethink many of the well-established notions of what a *project manager* is over some period of time. My opinion is that:

- Those changes will dramatically impact the role of project managers in many environments and perhaps even eliminate the role of some project managers as we have known them.
- It will also raise the bar for the entire project management profession, broaden the definition of what we think of as *project management,* and require project managers to acquire significant new skills and new ways of thinking.

Some people may see that as unsettling and perhaps even threatening; however, it is very clear that agile is not a fad, is here to stay, and will bring about some significant changes that we can't ignore. I believe that it is critical for project managers and the project management profession, as a whole, to be proactive and anticipate the most likely impact and adapt accordingly. To me, that means figuring out how to integrate agile and traditional project management principles and practices to provide one integrated view of what project management is.

Many project managers are wondering what impact this has on their career path and it can be confusing because the role of a *project manager* in an agile environment is not defined. This raises a number of questions including:

- What is the role for a project manager in an agile project?
- Are traditional project management principles and practices in conflict with agile principles and practices?
- How does a typical project manager shape his or her career to move in a more agile direction?

Those are the needs and challenges that this book is intended to address. Learning to become an agile project manager can be a long and difficult journey, and this book is only a small part of that journey.

LEARNING OBJECTIVES FOR AN AGILE PROJECT MANAGER

The following is a summary of what I believe are the most important steps in the journey toward becoming an agile project manager (not necessarily in this order):

1. Develop new ways of thinking and begin to see agile principles and practices in a new light as complementary rather than competitive to traditional project management practices.
2. Gain an understanding of the fundamentals of agile practices and learn the principles behind the agile practices at a deeper level in order to understand why they make sense and how they can be adapted as necessary to fit a given situation.

3. Learn how to go beyond the traditional notion of plan-driven project management and develop an adaptive approach to project management that blends both agile and traditional project management principles and practices in the right proportions to fit a given project and business environment.
4. Understand the potential roles that an agile project manager can play and begin to reshape project management skills around those roles.
5. Learn some of the challenges of scaling agile to an enterprise level and develop experience in applying these concepts in large, complex, enterprise-level environments.

HOW THIS BOOK IS ORGANIZED

Agile project management is an art that will take time for anyone to develop and master. There's a concept from martial arts called *shu-ha-ri* that is very appropriate here. It outlines the stages of proficiency someone goes through to develop mastery of martial arts techniques. The same concept can be applied to agile project management:

- **"Shu":** In the "shu" stage, the student learns to do things more-or-less mechanically, "by the book," without significantly deviating from the accepted rules and practices and without improvising any new techniques. This stage is equivalent to a new inexperienced project manager following PMBOK or other accepted practices "by the book" without necessarily adapting those practices to fit the situation.
- **"Ha":** In the "ha" stage, the student begins to understand the principles at a deeper level and learns how to improvise and break free from rigidly accepted practices, but it's important to go through the "shu" stage and gain mastery of the foundational principles before you start improvising—improvisation without knowledge is just amateurish experimentation.
- **"Ri":** Finally, in the "ri" stage, the student gets to the highest level of mastery and is able to develop his/her own principles and practices as necessary.

Many project managers may think that they are already at a very high level of mastery based on their knowledge of PMBOK and other well-accepted traditional project management practices, but agile changes that dramatically and raises the bar significantly.

The way the book is organized follows the *shu-ha-ri* approach to learning:

- The initial sections of the book start out with a very basic understanding of the "mechanics" of agile and learning how to do it "by the book." That is equivalent to the "shu" level of training.
- The book will go deeper into the principles behind agile and why they make sense. It is essential to understand the principles at a deeper level before moving on to the "ha" level and know how to customize an approach to fit a given situation.
- The final goal is to move to the master level or "ri" level where you will learn to go beyond current ways of implementing both agile and plan-driven approaches and learn how to blend them together

as needed to fit a given project and business environment. That goal is somewhat beyond the scope of this book and will only come from actual practice in implementing these ideas in real world situations; however, it is hoped that the information in this book and the case studies that are included will help project managers move rapidly in that direction.

Part 1 – Fundamentals of agile

The first step in learning to become an agile project manager is to learn the fundamentals of agile, which includes not only the mechanics of how an agile project based on Scrum works, but also understanding the principles behind it at a deeper level so that you can go beyond just implementing it by the book.

Part 2 – agile project management

Agile is causing us to broaden our vision of what a *project manager* is and that will have a dramatic impact on the potential roles that a project manager can play in an agile project. In fact, the role of a project manager at a team level in a typical agile/Scrum project is undefined. That will cause us to rethink many of the things we have taken for granted about project management for a long time to develop a broader vision of what an *agile project manager* is.

Part 3 – Making agile work for a business

There are many precedents for successful implementation of agile principles and practices at a project team level; however, extending the agile principles and practices to large-scale enterprise implementations and integrating with a business environment can be very difficult and introduces a number of new challenges, which include:

- Large, complex projects that are commonly found at an enterprise level may require some reinterpretation and adaptation of agile principles and practices as well as blending those principles and practices with traditional, plan-driven principles and practices in the right proportions.
- Integrating agile principles and practices with higher levels of management typically found at an enterprise level, such as project portfolio management and overall business management can be difficult. However, if an agile implementation is limited to a development process only and does not address integration with these higher-level processes it is not likely to be effective and may result in failure.
- This section of the book is intended to address these topics and provide an understanding of the key considerations that need to be addressed for scaling an agile approach to an enterprise level, integrating it with a business environment, and planning and implementing an enterprise-level agile transformation.

Part 4 – Enterprise-level agile frameworks

Putting together a complete, top-to-bottom, enterprise-level agile solution can be a very challenging task, especially when some of the pieces are not designed to fit together. To simplify the design of an enterprise-level agile implementation, it is useful to have some predefined frameworks that can be modified to fit a given business environment, rather than having to start from scratch to design an overall management approach. Three frameworks are discussed in this section: the Scaled Agile Framework (SAFe) (Dean Leffingwell), Managed Agile Development framework (Chuck Cobb), and the Disciplined Agile Delivery framework (Scott Ambler).

Part 5 – Case studies

In any book of this nature, it's always useful to go beyond theory and concepts and show how companies have actually put these ideas into practice in the real world. Of course, there is no canned approach that works for all companies—each of these case studies is different and shows how a different approach may be needed in different situations. It also includes a chapter on "Not-So-Successful" case studies, which shows some of the problems that can develop in an agile implementation.

Part 6 – Appendices

The appendices to the book include additional supplementary information:

- Additional Reading List
- Glossary of Terms
- Example Project/Program Charter
- Suggested Course Outline for a graduate-level course to accompany this book

ACKNOWLEDGMENTS

I USED A VERY AGILE APPROACH for writing this book as well as my previous books. It was a team effort of a number of people who worked with me collaboratively as the book was being written to provide feedback and inputs. I particularly want to thank the following people for their contributions to the book:

- Erik Gottesman, director general management at Sapient—Erik is a significant thought leader in this area. He played a huge role in helping me develop my two previous books on agile project management and provided some good advice and input on this book as well.
- Dr. Michael Hurst, PMO director at Harvard Pilgrim Health Care—Michael has played a significant role in providing input and advice for both this book and my last book and he also played a key role in providing a case study on Harvard Pilgrim Health Care that is included in this book.
- Andrew Bone, IT program/PMO director—Andrew did a thorough review of the entire book, provided a number of good comments and inputs, and also sponsored a presentation on the book with the Long Island, New York, PMI Chapter.
- Liza Wood, senior production manager at Warner Bros. Games—Liza also did a thorough review of the entire book on behalf of the PMI Agile Community of Practice and provided a very large number of excellent comments.
- Several companies generously shared case studies with the results of successful agile implementations:
  - Michael Hurst, director PMO, Harvard Pilgrim Health Care—Michael and Harvard Pilgrim shared the results of a very large and successful enterprise-level agile transformation effort of more than 200 projects.
  - Stephanie Stewart, director of agile leadership at Valpak—Stephanie and Valpak shared the results of an enterprise-level implementation of the Scaled Agile Framework at Valpak.
  - Nigel Edwards, program manager at General Dynamics, UK—Nigel shared the results of a very large and complex, agile fixed price government contracting effort.

I would like to also thank the following individuals who took the time to review an early draft of this book and provided very helpful feedback, comments, and suggestions:

Tanvir Ahmed, PMP, CSM,PSM	Sr. consultant—Agile process improvement and implementation	Philadelphia Water Department
Gopi Aitham, PMP, CSM, ITIL, SSGB	Learner, educator, & entrepreneur	
Chris Chan	Supervising consultant, enterprise agile coach	Object Consulting
David G Peterson, PMP	Consultant	
Czeslaw Szubert, PMP	Program manager	AMD
Kevin Wegryn, PMP, PfMP	Vice president	

1 Introduction to Agile Project Management

OVER THE PAST 10 TO 15 YEARS, there has been a rapid and dramatic adoption of agile methodologies:

1. Project Management Institute (PMI)® studies concluded that from 2008 to 2013, the use of agile practices tripled.[1]
2. According to a 2013 survey conducted by VersionOne:[2]
   - 88% of the respondents say that their organizations are practicing agile development, up from 84% in 2012 and 80% in 2011.
   - Over half of the respondents (52%) are using agile software to manage the majority of their projects.
   - 88% say that they are at least "knowledgeable" about agile software development techniques, up 7% from the previous year.
3. This trend has been going on for some time. As early as 2007, a Forrester survey reported:[3]
   - "26% are already using agile and an additional 42% are aware."
   - "Adoption of agile increased 56% from 17% in 2006, to 26% in 2007."
   - "Awareness increased 45% from 29% in 2006, to 42% in 2007."

These statistics indicate that agile is not a fad, it is having a significant impact on the way projects are managed, and it's definitely here to stay. This trend has a significant impact on the career direction of project managers who have come from a traditional, plan-driven project management background since there is no formal role for a project manager at the team level in an agile project.

[1] "Agile Project Management," Project Management Institute, 2014, http://learning.pmi.org/course-detail.php?id=2563
[2] "2013 State of Agile(TM) Survey," VersionOne, Inc., 2014, http://stateofagile.versionone.com/
[3] Rally Blogs, "Agile Adoption Rates—So What and Why Do I Care?" posted by Ryan Martens, March 6, 2008, www.rallydev.com/community/agile-blog/agile-adoption-rates-%E2%80%93-so-what-and-why-do-i-care.

THE CHASM IN PROJECT MANAGEMENT PHILOSOPHIES

In spite of this rapid and sustained proliferation of agile, there is still a fairly large chasm between the agile and traditional project management communities:

- There has been only a limited amount of progress on developing a more integrated approach to project management that embraces both agile and traditional plan-driven project management principles and practices.
- Many people seem to see agile and project management principles and practices as competitive approaches that are in conflict with each other, and they are essentially treated as two separate and independent domains of knowledge.
- Considerable polarization between these two communities is based in some part on myths, stereotypes, and misconceptions about what *agile* and *project management* is.

A major goal of this book is to help project managers understand the impact of agile on the project management profession and to broaden and expand their project management skills as needed to develop a more integrated approach to adapt to this new environment.

This isn't just a matter of getting another certification—it can require a major shift in thinking for many traditional project managers that will take time and experience to develop. PMI has created a new PMI-ACP® (Agile Certified Practitioner) certification, which has been very successful and is a great step in the right direction—but it doesn't go far enough, in my opinion. It doesn't test whether a project manager knows how to blend agile and traditional project management principles and practices in the right proportions to fit a given situation, and that is the real challenge that many project managers face.

A lot of the polarization that exists between the agile and traditional project management communities is rooted in some well-established stereotypes of what a *project manager* is that are based on how typical projects have been managed in the past. The role of a project manager has been so strongly associated with someone who plans and manages projects using traditional, plan-driven project management approaches that many people can't conceive of any other image of a project manager. It's time to develop a new vision of what an *agile project manager* is that goes beyond all of those traditional stereotypes and fully integrates *agile* within the overall portfolio of project management principles and practices.

It feels very similar to an evolution that took place when I worked in the quality management profession in the early 1990s. Up until that time, the primary emphasis in quality management had been on *quality control,* and inspection, and the image of a *quality manager* was heavily based on that role:

- The predominant quality management approach was based on final inspection of products prior to shipping them to the customer and rejecting any that didn't meet quality standards. It's easy to see how that approach was inefficient, because it resulted in a lot of unnecessary rework to correct problems after the fact, and it also wasn't that effective because any inspection approach is based on sampling, and it is impractical to do a 100% sample. For that reason, it can result in mediocre quality.

- A far better approach was to go upstream in the process and eliminate defects at the source by designing the process to be inherently more reliable and free of defects and build quality into the inherent design of the products. That didn't mean that the prior emphasis on quality control and inspection was obsolete and eliminated; it was just not the *only* way to manage quality and wasn't the most effective approach in all situations.

That was a gut-wrenching change for many in the quality management profession—instead of being in control of quality and being the gatekeeper with the inspection process, a good quality manager needed to become more of a coach and a consultant to influence others to build quality into the way they did their work. This changed the nature of the work dramatically for many in the quality management profession and eliminated a number of traditional quality management roles that were based on the old quality control and inspection approach. The similarity to the changes going on in the project management profession should be apparent:

- To be successful in more uncertain environments, project managers need to be able to take an adaptive approach that is appropriate to the level of uncertainty in the project and integrate quality into the process rather than relying on final acceptance testing at the end of the project to validate the product that is being produced.
- They also need to give up some of the control that has become associated with the project management profession—in some cases, they may need to become more of a coach and a consultant to influence others rather than being in absolute control of a project.

This can dramatically change the role of a project manager. In some situations, the role of a project manager as we've known it may no longer exist. For example, at a team level in an agile project, you probably won't find anyone with a title of *project manager* because the project management functions have been absorbed into other roles and are done very differently. That doesn't mean that *project management* is no longer important, but it may cause us to dramatically rethink what project management is in a much broader context than the way we might have thought about it in the past.

THE EVOLUTION OF AGILE AND WATERFALL

You will often hear people make a comparison between agile and waterfall. Many of those discussions are polarized and position them as competitive approaches. Here's an example:[4]

> According to the 2012 CHAOS report, Agile succeeds three times more often than Waterfall. Because the use of Agile methodologies helps companies work more efficiently and deliver winning results, Agile adoption is constantly increasing.

[4] "Agile Adoption Statistics 2012," One Desk, May 16, 2013, http://community.spiceworks.com/topic/337418-agile-adoption-statistics-2012.

While that statement is generally true, it's an oversimplification. There are at least two problems with that kind of statement: a

1. It makes it sound like there are only two binary, mutually exclusive choices, agile and waterfall.
2. The meaning of the words *agile* and *waterfall* are typically not well-defined and are used very loosely.

For those reasons, I prefer to avoid comparing agile to waterfall because it tends to be a very polarized discussion—I prefer to take a more objective approach that is based on a comparison between a plan-driven and an adaptive approach to project management. So let's first define both *agile* and *waterfall,* and then compare the two approaches.

Definition of waterfall

The word *waterfall* actually has a very specific meaning, but that's often not how the word is really used:

> The waterfall model is a popular version of the systems development life cycle model for software engineering. Often considered the classic approach to the systems development life cycle, the waterfall model describes a development method that is linear and sequential. Waterfall development has distinct goals for each phase of development. Imagine a waterfall on the cliff of a steep mountain. Once the water has flowed over the edge of the cliff and has begun its journey down the side of the mountain, it cannot turn back. It is the same with waterfall development. Once a phase of development is completed, the development proceeds to the next phase and there is no turning back.[5]

Another aspect to the waterfall model is that it is plan-driven; it attempts to define and document detailed requirements and a plan for the entire project prior to starting the project. When someone makes a statement comparing waterfall to agile, the word *waterfall* is often used very loosely to refer to any kind of plan-driven methodology, and that's not really a very accurate and meaningful comparison. In some other comparisons like this, the word *waterfall* refers to a general style of project management that obsessively emphasizes predictability and control over agility, and that's just bad project management.

Definition of agile

The meaning of the word *agile* in this kind of comparison is also somewhat elusive because it has taken on some very strong connotations in actual usage. At a project level, at least in the

[5]Margaret Rouse, "Waterfall Model," TechTarget, February 2007, http://searchsoftwarequality.techtarget.com/definition/waterfall-model.

United States, the word *agile* has taken on a specific connotation associated with using the Scrum methodology on software development projects:

> Scrum is an agile software development model based on multiple small teams working in an intensive and interdependent manner. The term is named for the scrum (or scrummage) formation in rugby, which is used to restart the game after an event that causes play to stop, such as an infringement. Scrum employs real-time decision-making processes based on actual events and information.[6]

That definition has evolved over recent years as Scrum has become somewhat of a de-facto standard for agile projects; however, the original definition of *agile* conceived in the *Manifesto for Agile Software Development*, published in 2001, was much broader than that. Better known as the Agile Manifesto, it laid out some simple and general principles and values that can apply to any kind of project (not just software development) (See Chapter 2).

Comparison of plan-driven and adaptive approaches

Traditional, plan-driven project management is a style of project management that is applied to projects where the requirements and plan for completing the project can be defined to some extent prior to implementing the project. However, *plan-driven* is a relative term, and you won't find many projects that start out with an absolutely rigid plan that is not expected to change at all.

In contrast, an adaptive style of project management starts the implementation of a project with a less well-defined plan of how the project will be implemented and recognizes that the requirements and plan for the project are expected to evolve as the project progresses. *Adaptive* is also a relative term; you won't find many projects that have no plan whatsoever of how the project will be done.

The important point is that the terms *plan-driven* and *adaptive* are relative—they are not discrete, binary, mutually exclusive alternatives. They should imply a continuous range of approaches with different levels of upfront planning.

Saying "Agile is better than waterfall," is like saying, "A car is better than a boat." Agile and waterfall are different kinds of methodologies designed for different kinds of projects. The problem is not so much that waterfall or agile are inherently good or bad; the problem comes about when those methodologies are misused and people try to use a single methodology (whatever it might be) for all projects. Using a "one size fits all" strategy to applying either waterfall (plan-driven) or agile (adaptive) approaches to all projects is not likely to yield optimum results.

In my opinion, being able to objectively understand the difference between a plan-driven approach and a more adaptive approach—as well as the principles behind those approaches at a

[6] Margaret Rouse, "Scrum," TechTarget, February 2007, http://searchsoftwarequality.techtarget.com/definition/Scrum.

deeper level—is probably one of the most important skills an agile project manager needs to have. An agile project manager needs to recognize the following:

- *There is a broad range of alternative approaches between being plan-driven and being adaptive*. The agile project manager must choose the right level of upfront planning to be applied to a project, based on the level of uncertainty and other factors in the project.
- *It takes some skill to make the right choice*. There is nothing inherently wrong with either of those approaches (adaptive or plan-driven). The problem comes about when people try to force-fit a project into one of these approaches rather than selecting and tailoring the approach to fit the project. For example, if I were to set out to try to find a cure for cancer and I attempted to apply a highly plan-driven approach to that project, the results would probably be dismal.

The important point is that a heavily plan-driven approach (what some loosely refer to as *waterfall*) is not the only way to successfully manage a project. In many projects, a good approach is to use an adaptive approach to start the design effort without fully-defined and detailed requirements and perhaps prototype something quickly. Then user feedback can be added to further refine the design as the project progresses. With a more adaptive approach:

- *The elements of the approach are much more concurrent than sequential*. Instead of doing the entire design and then turning it over to quality assurance (QA) for testing, the design is done in small chunks called *iterations* or *sprints* that are typically two to four weeks long. During that time, developers and testers work collaboratively to design and test the software during each sprint.
- *The customer also provides detailed inputs on the design during each sprint*. The customer accepts the results of each sprint at the end of each two- to four-week period rather than waiting for user acceptance testing (UAT) at the end of the project. That has the advantage of finding and resolving any problems quickly and early in the project.

One primary advantage of a more adaptive approach is that the project startup is accelerated because less time is spent upfront in attempting to define detailed requirements. In addition, engaging the user more directly in the design process is more likely to produce an outcome that provides the necessary business value and really meets the user needs.

An adaptive approach maximizes the business value to the customer because the customer is directly engaged with the design team as the project progresses, but it is worse for predictability and control because the customer can make changes as the project progresses. In an agile project, change is the norm rather than the exception. However, this is not an "all-or-nothing" proposition to have total control or no control at all. There are many ways to achieve the right balance of control versus agility. For example, prior to the start of a project, the high-level requirements might be defined and stabilized, and then only the more detailed requirements need to be further elaborated as the project progresses.

THE EVOLUTION OF THE PROJECT MANAGEMENT PROFESSION

Many of the techniques associated with project management that are in use today haven't changed significantly since the 1950s and 1960s. I believe that we are on the verge of a major transformation of the project management profession that will cause us to redefine project management in a much broader context that includes both agile and traditional, plan-driven project management.

The early history of project management

In order to understand this transition and to put it in perspective, it is useful to understand how the project management profession has evolved over the years and how we got to where we are today. Project management has been practiced for many years in one way or another—I'm sure that there was some kind of "project management" approach for building the great pyramids of Egypt or the Great Wall of China or other similar large efforts many years ago, but it probably wasn't even thought of as project management in those days. They didn't have Gantt charts and Pert charts and other sophisticated project planning and management tools, because those things weren't even invented until the twentieth century.

The Industrial Revolution created the need for a more disciplined approach to project management, and a well-defined body of knowledge associated with project management began to evolve:

> In the late 19th century, in the United States, large-scale government projects were the impetus for making important decisions that became the basis for project management methodology such as the transcontinental railroad, which began construction in the 1860s. Suddenly, business leaders found themselves faced with the daunting task of organizing the manual labor of thousands of workers and the processing and assembly of unprecedented quantities of raw material.
>
> Near the turn of the century, Frederick Taylor began his detailed studies of work. He applied scientific reasoning to work by showing that labor can be analyzed and improved by focusing on its elementary parts that introduced the concept of working more efficiently, rather than working harder and longer.
>
> Taylor's associate, Henry Gantt, studied in great detail the order of operations in work and is most famous for developing the Gantt [c]hart in the 1910s.[7]

World War II brought about the need for more large-scale project management for organizing very large projects like the Manhattan project; however, it wasn't until the 1950s and 1960s, that it became apparent that a much more well-defined body of knowledge and a disciplined approach were

[7] PM Hut, "History of Project Management," The Project Management Hut, June 6, 2011, http://www.pmhut.com/history-of-project-management.

needed to successfully manage some of the large and complex projects that were evolving at that time, which led to the following:

- The Program Evaluation and Review Technique or PERT was developed by Booz-Allen & Hamilton as part of the US Navy's (in conjunction with the Lockheed Corporation) Polaris missile submarine program."
- The Critical Path Method (CPM) was developed in a joint venture between DuPont Corporation and Remington Rand Corporation for managing plant maintenance projects.
- The Project Management Institute (PMI) was founded in 1969.[8]

Many people probably assume that the project management profession is now reaching a stage of maturity and stabilizing, but I believe that we have only seen the beginning, and project management, as we've known it, will continue to grow in entirely new directions.

Transformation of the project management profession

Sometimes we get so immersed in day-to-day activities that we don't take time to step back and see some fundamental changes that are going on around us. It seems clear to me that the project management profession, as we know it, is going to go through such a major transformation. The exact nature of that transformation isn't completely clear as it is still evolving; however, it does seem likely that it will cause us to rethink many of the things we have taken for granted in the project management profession for a long time in a much broader perspective. It feels very similar to the evolution that has taken place in other technology areas and disciplines. For example, there is a strong similarity to the evolution from classical physics to modern physics.

> "By the close of the 19th century, the study of physics was widely thought to be essentially complete, with the exception of only a few 'loose ends'—minor unsolved problems to be dealt with."[9]

Up until that time, the study of physics had been heavily dominated by Newtonian physics, which defines some fundamental laws of how the universe behaves such as Newton's laws of motion. These fundamental laws have been taken for granted in the world of physics for many years, even though we didn't fully understand why things in the universe behaved as they did. As modern physics has evolved

[8]Ibid.
[9]Physics", http://www.conservapedia.com/Physics

in the twentieth century based on quantum mechanics and relativity, we began to develop a deeper understanding of the real dynamics behind these laws and we began to understand that the universe is not as simple and deterministic as we might have thought it was.

The transition from classical, Newtonian physics to a more complete and more dynamic model based on quantum mechanics provided a deeper understanding of the forces and principles behind those laws as well as the limitations in those laws and when and where they are really applicable. That deeper understanding didn't invalidate the laws of Newtonian physics in most situations—"on an 'everyday' scale; that is, situations in which energies are large enough to permit one to neglect quantum effects, but small enough to neglect relativistic effects."[10]

The similarity to the transition in the project management profession should be apparent—we're moving from a world in which we had the impression that the behavior of the universe was highly predictable and controllable and totally subject to some well-defined rules to a world that is much more dynamic, much more probabilistic, and much less predictable.

What's driving this change, and why now?

You might ask, "Why is it becoming so essential for the project management profession to make a change at this particular point in time? There are several major factors that will force us to rethink the concept of project management:

1. *The nature of projects is changing*. The modern concepts of project management were developed as result of big projects like the transcontinental railroad . . . Today, we have new industries and a much broader range of projects such as web development, e-commerce, large IT projects, etc., which weren't common before the mid-nineties. It is becoming increasingly apparent that applying a "one size fits all" approach to such a broad range of projects will not have optimum results.
2. *Technology is rapidly changing*. Figure 1.1 shows how the adoption rate of new technologies has changed over the past century. Project management approaches that worked in the 1950s and 1960s must be reexamined to adapt to the current fast pace of technology adoption.

A similar transformation took place in the quality management profession in the 1980s and early 1990s. At that time, the Japanese auto industry was demonstrating huge improvements in quality of products that made conventional quality management methods based primarily on quality control and inspection very inadequate. They forced people to rethink the whole strategy and approach for doing quality management. Without the leadership of people like W. Edwards Deming and the significant improvements in quality that were demonstrated in the automotive industry, the transformation of quality management might never have happened. What started primarily in the automotive industry

[10] "Physics", http://www.conservapedia.com/Physics

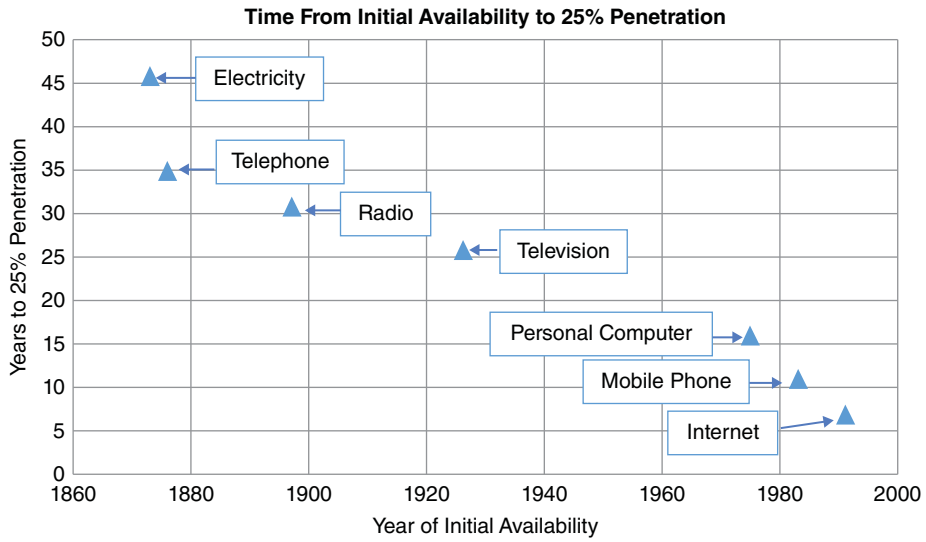


FIGURE 1.1 Adoption rates of new technologies

Created with data from singularity.com

has now become a more modern approach to quality management that is widely used in all industries. A similar thing is happening with agile and traditional project management today:

- The leadership of W. Edwards Deming in establishing a total quality management (TQM) philosophy can be compared to the thought leadership behind the Agile Manifesto in 2001.
- The broad-based adoption of TQM starting in the automotive industry and eventually spreading to many other industries can be compared to how agile has started out in software development today and is now beginning to spread to other areas.

Other researchers have come to a similar conclusion regarding this; Manfred Saynish published his findings of a research project in *Project Management Journal*:

> Traditional Project Management... is based mainly on a mono-causal, non-dynamic, linear structure and a discrete view of human nature and societies and their perceptions knowledge and actions. It works on the basis of reductionist thinking and on the Cartesian/Newtonian concept of causality (the mechanistic science).[11]

[11] Manfred Saynish, "Mastering Complexity and Changes in Projects, Economy, and Society via Project Management Second Order (PM-2)," *Project Management Journal* 41, no. 5 (Dec. 2010).

The article proposes a new approach to project management called "PM-2" where traditional project management will play an active and important role but will be "extended to consider dynamic, nonlinear, multi-causal structures and processes as well as the principles of self-organization, evolution, and networking." The article goes on to say:

> For an effective attainment of project goals at a defined finishing point in time, we need the linear processes and Cartesian causality and the Newtonian logic from Traditional project management. But evolutionary and self-organizational-based management methods are necessary to master complex and uncertain situations on the way to the defined finishing point in time for a project. A well-balanced interaction of traditional project management and the evolutionary and self-organizational principles is the message of the Project Management Second Order.[12]

I agree with that view—we are on the verge of a new generation of project management that will cause us to rethink many of the accepted notions of what "project management is." It requires blending traditional project management principles and practices with a much more empirical and evolutionary approach to deal with the uncertain, dynamic, and fast-paced environment we live in today.

AGILE PROJECT MANAGEMENT BENEFITS

I am a strong believer in agile, and there are some very significant benefits of an agile approach in many situations. However, many proponents of agile oversell the benefits and sometimes position agile as a panacea that should be used for all projects. The real benefit to a typical project manager of developing an agile project management approach is *not* in throwing away any notion of using a plan-driven approach, converting to agile, and using a totally agile approach for all projects. Rather, the benefit results from recognizing that a traditional, plan-driven approach is not the best way to manage *all* projects and thus learning to blend adaptive/agile and plan-driven principles and practices in the right proportions to fit a given situation.

Even if a project manager never uses a *fully* agile approach, I believe that knowledge of agile concepts and principles will make him/her a better project manager. It's really a matter of learning a broader range of approaches (adding more tools to your tool box) and developing a more adaptive project management approach (developing more skill in using those tools). In my previous books,

[12]Ibid.

I used the analogy of a project manager as a "cook" and the project manager as a "chef" (with credit to Bob Wysocki):[13]

- A good *cook* might have the ability to create some very good meals, but those dishes might be limited to a repertoire of standard dishes, and his/her knowledge of how to prepare those meals might be primarily based on following some predefined recipes out of a cookbook.
- A *chef,* on the other hand, typically has a far greater ability to prepare a much broader range of more sophisticated dishes using much more exotic ingredients in some cases. The chef's knowledge of how to prepare those meals is not limited to predefined recipes, and in many cases, a chef will create entirely new and innovative recipes for a given situation. The best chefs are not limited to a single cuisine and are capable of combining dishes from entirely different kinds of cuisine.

I think that sums up the transformation that needs to take place—we need to develop more project managers who are "chefs" rather than "cooks."

Here are five specific benefits of developing an agile project management approach:

1. *Increased focus on business outcomes.* Many people think that the primary benefit of an agile project is just getting it done faster, but that is not always the case. The primary emphasis in an agile project is really to deliver value in the form of very successful business outcomes by taking an adaptive approach to maximize the value that is delivered. That doesn't always result in the fastest delivery times. In some cases, it may require some experimentation and trial-and-error prototyping to find an optimum solution—that may or may not be the quickest way to get it done, but it should result in a better product in the end.
2. *Reduced time to market.* Time to market is, of course, an important consideration, and agile accomplishes that in a couple of ways:
   - By reducing the startup time required for projects as a result of simplifying some of the requirements definition practices
   - By improving the efficiency of the overall project and delivering functionality incrementally as much as possible
   - By focusing on simplicity and eliminating non–value-added work
3. *Higher productivity and lower costs.* Agile can also result in higher productivity and lower costs by eliminating unnecessary overhead and bottlenecks and doing work concurrently rather than sequentially.
4. *Higher quality.* A very important benefit of agile is higher quality. In a traditional waterfall project, *quality* is sequential and is often perceived as a separate effort that is the responsibility of the quality assurance (QA) department. The developers many times develop the software and then "toss it over the wall" to be tested by QA. In an agile project, the team, as a whole (which includes QA testers) jointly owns responsibility for building quality into the design of

[13]Cobb, Charles, Making Sense of Agile Project Management, Wiley, 2003, p 96

the products that they produce—it's not someone else's responsibility. The development effort is broken up into short iterations called *sprints* that are typically two to four weeks in length. There is an emphasis on producing a *shippable product* at the end of every sprint, which means that quality testing must be more integrated with development and cannot be put off indefinitely.

In the traditional environment, the developers may pass software over to QA that has not been fully tested, expecting QA to test it and find any bugs. In an agile environment, code is not considered "done" until it has been tested and proven to be working without defects.

5. *Organizational effectiveness*. Finally, a very important benefit of agile is a more effective organization with higher morale:
   - People at all levels are motivated and empowered to do their work and take pride in doing it well because the environment is built on solid values, including respect for people.
   - All parts of the organization work together more collaboratively in a spirit of partnership toward common goals.

SUMMARY OF KEY POINTS

1. *Closing the chasm*. There has been a widespread and rapid adoption of agile methodologies over the last 10 to 15 years; however, our progress in developing an integrated approach to project management that embraces agile as well as traditional project management principles and practices has been somewhat limited. To make progress in that direction, we need to get past a number of well-established stereotypes and develop a much broader vision of what project management is.
2. *Comparison of agile versus waterfall*. The typical discussion that compares agile and waterfall as if they were two discrete, mutually exclusive, binary choices oversimplifies what should be more accurately thought of as a range of adaptive and plan-driven approaches. The agile versus waterfall comparison has also created an impression that the approaches are competitive, and that has created some polarization. In fact, adaptive and plan-driven approaches really should be thought of as much more complementary to each other.
3. *Transformation of the project management profession*. The project management profession is at a major turning point in its history. The project management profession has developed over a number of years into a well-planned and disciplined approach to how projects are managed in reaction to the need for managing very large and complex projects that evolved in the early 1950s and 1960s. That approach has worked well for projects that can be heavily plan-driven; however, it has serious limitations in highly uncertain and rapidly changing environments that are difficult or impractical to plan.
4. *Agile project management benefits*. Developing a more adaptive approach to project management and tailoring the approach to fit the project will generally result in a number of benefits. The

benefits come from matching the approach to the project rather than always using a plan-driven approach for all projects. These benefits are not limited only to agile projects—even if a project manager is never involved in an Agile project at all, developing a broader and deeper knowledge of both adaptive and plan-driven principles and practices is likely to significantly improve a project manager's skills for many different projects by developing a more adaptive approach that can be optimized for the nature of the project.

DISCUSSION TOPICS

1. Closing the Chasm

Have you observed the "chasm" between the agile and traditional project management communities? How does it manifest itself? What is the impact? What needs to be done to "close the chasm"?

2. Agile versus Waterfall

Research the usage of the terms *agile* and *waterfall*. Identify and discuss how this comparison is often misleading. Explain the difference between an *adaptive* and *plan-driven* approach and how that helps to provide a more objective frame of reference.

3. Transformation of the Project Management Profession

How do you see the transformation that is going on in the project management profession? Do you think that there is a significant change, and if so, what impact will it have on the project management profession as a whole? What needs to be done to make this transformation happen?

4. Agile Project Management Benefits

What do you think are the most important benefits of developing a more adaptive/agile approach? How would it affect the way you manage projects?

5. Balancing Agility and Control

How would you go about determining the appropriate balance of agility and control for a project? What factors would you consider, and why? Provide an example of a real-world project and discuss how you might do it differently based on these factors.

6. Agile Benefits

What do you think are the most important benefits of an agile approach to some typical projects? Discuss a real-world example of how an organization might have benefited from adopting a more agile approach.

7. Project Management Career Direction

How do you think agile affects the career direction of project managers? What impact do you think it might have on your own career? What do you think you might have to do differently as a result of agile?

PART 1

Fundamentals of Agile

THE FIRST STEP IN LEARNING to become an agile project manager is to learn the fundamentals of agile, which includes understanding not only the mechanics of how an agile project based on Scrum works but also the principles behind it at a deeper level so that you can go beyond just implementing it mechanically, by the book.

Chapter 2 – Agile History and the Agile Manifesto

Agile is based on the values and principles expressed in the Agile Manifesto that was originally created in 2001. The values and principles in the Agile Manifesto were a reaction to many of the existing project management approaches at that time that were perceived to be overly prescriptive, bureaucratic, and not very effective in a software development environment where the level of uncertainty is typically very high.

Chapter 3 – Scrum Overview

Scrum has become, by far, the most widely used agile methodology in the United States, and is rapidly being used in other areas of the world as well. In fact, Scrum is so widely used that it means *agile* to many people, just as "Coke" is sometimes used generally for carbonated soda drinks. Although Scrum is most heavily used in a software development environment, it provides a framework that can be easily adapted to a wide range of projects not limited to software development. An understanding of how Scrum works is essential for any agile project manager, since it embodies all of the values and principles of the Agile Manifesto.

Chapter 4 – Agile Planning, Requirements, & Product Backlog

Probably the most important area of any agile process, and the area that is most significantly different from traditional plan-driven projects, is the area of how planning is done and how requirements are defined and managed. An understanding of these differences is probably one of the most critical areas for an agile project manager to understand.

Chapter 5 – Agile Development, Quality, and Testing Practices

You might ask: "Why does a project manager need to know something about development practices?" In the past, the role of a project manager might have been somewhat limited to a coordination function to integrate the efforts of the different functional organizations that played a role in the project. In many cases, the actual functional direction for the different functions involved (development, test, etc.) came from the managers who were responsible for those functions themselves, and the role of the project manager in providing functional direction may have been limited. An agile environment is different—cross-functional teams are much more integrated, and an agile project manager needs to be a strong, cross-functional leader.

2 Agile History and the Agile Manifesto

AGILE EARLY HISTORY

THE EARLY HISTORY OF AGILE was influenced by a number of different trends, including:

- The evolution of new concepts in manufacturing, such as just-in-time and lean manufacturing processes
- New approaches to quality management, including the total quality management movement developed by Dr. W. Edwards Deming

Those fundamental trends will be discussed in Chapter 11, "Understanding Agile on a Deeper Level"; however, the strongest factor that has driven the agile movement is the unique risks and challenges associated with large, complex software development projects. I can remember a time when a computer that had 16 KB (kilobytes) of memory and a 5-MB (megabyte) disk was a lot. Even before that, I can remember the old days of developing programs on punched cards and paper tape, or even toggling in some binary, assembly language code into the switch register of an early Digital Equipment Corporation (DEC) mini-computer.

In those early days, software development was very limited and primitive; however, computer technology has changed rapidly and significantly since that time. Today, my iPhone 5 Smartphone has thousands of times more power than some of those early computers I worked with in the 1980s, and that has enabled the development of much more powerful and complex software to utilize that additional processing power. As software development has become much more widespread and has grown into much larger and more complex applications, it has created a number of new challenges for managing large, complex software development projects and a number of different software development methodologies have evolved over the years to meet these new challenges.

Note

While software development has been a key driving force behind agile and is still heavily associated with agile today, it should be clearly understood that agile is not limited to software development and is rapidly spreading to other areas. Any area that has a high level of uncertainty that cannot be easily resolved and/or high levels of complexity is probably a good candidate for an agile approach.

Dr. Winston Royce and the Waterfall model (1970)

In 1970, Dr. Winston Royce published a famous paper on "Managing the Development of Large Software Systems."[1] That paper is widely associated with the waterfall approach as we know it today and outlined an approach for breaking up a large complex software project into sequential phases to manage the effort. The model Dr. Royce proposed in his original 1970 paper is shown in Figure 2.1:

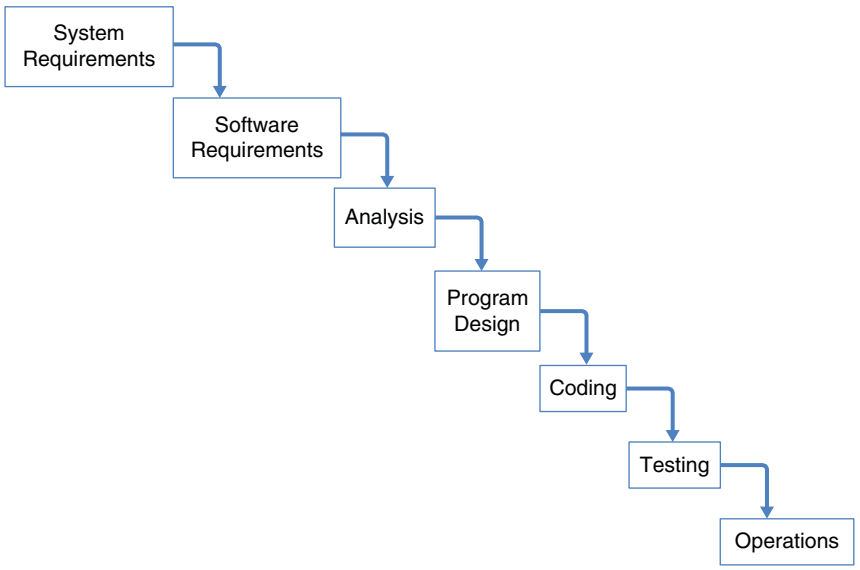


FIGURE 2.1 Original Waterfall model

[1]Dr. William Royce, "Managing the Development of Large Software Systems," ***Proceedings, IEEE Wescon*** **(August 1970), pp. 1–9.**

However, even Dr. Royce recognized the weaknesses in this approach at the time:

> I believe in this concept, but the implementation described above is risky and invites failure. The problem is illustrated in Figure [2.1]. The testing phase which occurs at the end of the development cycle is the first event for which timing, storage, input/output transfers, etc., are experienced as distinguished from analyzed. These phenomena are not precisely analyzable. They are not the solutions to the standard partial differential equations of mathematical physics for instance. Yet if these phenomena fail to satisfy the various external constraints, then invariably a major redesign is required. A simple octal patch or redo of some isolated code will not fix these kinds of difficulties. The required design changes are likely to be so disruptive that the software requirements upon which the design is based and which provides the rationale for everything are violated. Either the requirements must be modified, or a substantial change in the design is required. In effect the development process has returned to the origin and one can expect up to a 100-percent overrun in schedule and/or costs.[2]

One of the fundamental problems with the waterfall approach that Dr. Royce recognized was that the entire process was sequential and an error or omission made in one of the very early phases of the project may not have been discovered until the very end of the project which might require huge amounts of rework to the work that had already been done in the early phases of the project. For that reason, more iterative approaches began to evolve that featured more incremental and evolutionary development approaches.

Early iterative and incremental development methods (early 1970s)

Walker Royce, the son of Dr. Winston Royce and a contributor to the development of iterative and incremental development (IID) methods in the 1990s, made the following comment regarding his father's thinking:

> He was always a proponent of iterative, incremental, evolutionary development. His paper described the waterfall as the simplest description, but that it would not work for all but the most straightforward projects. The rest of his paper describes [iterative practices] within the context of the 60s/70s government contracting models (a serious set of constraints).[3]

[2]Ibid., p. 2.
[3]Vic Basili and Craig Larman, *Iterative and Incremental Development: A Brief History* (IEEE Computer Society Digital Library, June 2003), http://www.craiglarman.com/wiki/downloads/misc/history-of-iterative-larman-and-basili-ieee-computer.pdf, p. 2.

Many of the earliest efforts to break up large projects into incremental and iterative development efforts focused primarily on breaking up the development phase into iterations.

> The next earliest reference in 1970 comes from Harlan Mills, a 1970s software engineering thought leader who worked at the IBM FSD. In his well-known "Top-Down Programming in Large Systems" Mills promoted iterative development. In addition to his advice to begin developing from top-level control structures downward, perhaps less appreciated was the related life-cycle advice Mills gave for building the system via iterated expansions . . . .
>
> Clearly, Mills suggested iterative refinement for the development phase, but he did not mention avoiding a large up-front specification step, did not specify iteration length, and did not emphasize feedback and adaptation-driven development from each iteration. He did, however, raise these points later in the decade.[4]

Further evolution of iterative and incremental development (mid- to late 1970s)

Support for iterative and incremental development continued to grow in the 1970s. In 1976, Mills wrote: "Software development should be done incrementally, in stages with continuous user participation and replanning, and with design-to-cost programming within each stage."[5]

Larman and Basili write that in the context of a three-year inventory control system, Mills went on to challenge the idea and value of up-front requirements or design specifications. Mills said:

> There are dangers, too, particularly in the conduct of these [waterfall] stages in sequence, and not in iteration—i.e., that development is done in an open loop, rather than a closed loop with user feedback between iterations. The danger in the sequence [waterfall approach] is that the project moves from being grand to being grandiose, and exceeds our human intellectual capabilities for management and control.[6]

Early agile development methods (1980s and 1990s)

During the 1980s and early 1990s, there was a proliferation of approaches designed to further improve the methodologies for large, complex software development projects:

> Agile methods were direct spinoffs of software methods from the 1980s, namely Joint Application Design (1986), Rapid Systems Development (1987), and Rapid Application Development (1991). However, they were rooted in earlier paradigms, such as Total

[4]Ibid., p. 3.
[5]Ibid., p. 4.
[6]Ibid., p. 4.

Quality Management (1984), New Product Development Game (1986), Agile Leadership (1989), Agile Manufacturing (1994), and Agile Organizations (1996) . . . ."[7]

Agile methods formally began in the 1990s with Crystal (1991), Scrum (1993), Dynamic Systems Development (1994), Synch-n-Stabilize (1995), Feature Driven Development (1996), Judo Strategy (1997), and Internet Time (1998). Other agile methods included New Development Rhythm (1989), Adaptive Software Development (1999), Open Source Software Development (1999), Lean Development (2003), and Agile Unified Process (2005). However, the popularity of Extreme Programming (1999) was the singular event leading to the unprecedented success of agile methods by the early 2000s."[8]

Another major influence during this same period of time was the evolution of the Rational Unified Process (RUP), which became another widely used iterative approach. (Variations later emerged from that, including the Enterprise Unified Process (EUP) in 2003 and the Agile Unified Process (AUP) in 2005).[9]

By the early 2000s, there was a broad and confusing proliferation of different methodologies. Ultimately, however, they evolved into a much more cohesive agile approach.

AGILE MANIFESTO (2001)

Many people point to the *Manifesto for Agile Software Development,* or the Agile Manifesto, published in 2001, as the true beginning of the agile movement today. The Agile Manifesto clearly condensed all of the earlier agile methodologies into a set of clearly-defined values and principles that are still valid today. "The whole fuss started with the meeting in Feb. 2001 at Snowbird Ski Resort. 17 of us met to discuss whether there was a common, underlying basis for our work in the 1990s around what had been referred to as 'light-weight processes.' None of us liked the term 'light-weight,' feeling it was a reaction against something, instead of something to stand for."[10]

One of the most important things to recognize about the Agile Manifesto and the values and principles behind it is that they must be interpreted in the context of whatever project they're applied to. The Agile Manifesto consists of four values and a number of principles that back up those values. The value statements are not meant to be absolute statements at all, and the Agile Manifesto clearly

[7]David F. Rico, "The History, Evolution and Emergence of Agile Project Management Frameworks," ProjectManagement.com (February 4, 2013), http://davidfrico.com/rico-apm-frame.pdf, p. 1.
[8]Ibid., p. 1.
[9]Scott W. Ambler, "History of the Unified Process," Enterprise United Process (2013), http://www.enterpriseunifiedprocess.com/essays/history.html
[10]Blogroll, "10 Years of the Agile Manifesto—It started in 2001 with the Manifesto," posted by Alistair on January 2, 2010, http://10yearsagile.org/it-started-in-2001-with-the-manifesto.

states that; however, in spite of that, many people have made the mistake of dogmatically treating these statements as absolutes. Think about these statements in a broad sense and interpret them in the context of a particular project. If you take that approach, most of these values and principles make sense to be applied to almost any project, not just "agile" projects.

Agile Manifesto values

The Agile Manifesto values that were published originally in 2001 are:

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> - Individuals and interactions over processes and tools
> - Working software 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.[11]

Individuals and Interactions over Process and Tools

The first statement indicates a preference for "individuals and interactions over process and tools." This statement is essentially a response to "command-and-control" project management practices that had been perceived as very impersonal and insensitive to people and rigidly defined processes where the "process manages you." From a project management perspective, it calls for a softer leadership approach with an emphasis on empowering people to do their jobs as well as flexible and adaptive processes, rather than a rigid, control-oriented management approach that is highly directive. Agile processes generally depend very heavily on empowered people making intelligent decisions and the power of collaborative teamwork.

The last part of this statement regarding "process and tools" might imply that there is no place in an agile project for process and tools, but that is certainly not the case:

- Agile uses well-defined processes like Scrum (see Chapter 3), but they are meant to be interpreted and implemented intelligently rather than followed rigidly and mechanically, and they are also designed to be adaptive rather than prescriptive.
- Tools can also play a supporting role, but the important thing is to keep the tools in the right context—they should be there to leverage and facilitate human interactions, not to replace them.

[11] http://agilemanifesto.org/.

Working Software over Comprehensive Documentation

The second value statement indicates a preference for "working software over comprehensive documentation." This statement was essentially a response to typical phase-gate project management processes that called for extensive documentation deliverables at the end of each phase. There was entirely too much emphasis on producing documentation, to the extent that the documentation took on a life of its own, and there was insufficient emphasis on producing working software.

Note: Although the Agile Manifesto was written around software development, there's really no reason why most of the values and principles can't be extended to other products if they are used intelligently in the context of whatever products they are applied to. For example, this value could read "working products over comprehensive documentation."

One of the problems with documentation is that it can inhibit normal communication. The typical waterfall project was heavily based on documentation. The project team would develop an elaborately detailed requirements specification, and the software would be tested against meeting that specification. In many waterfall projects, the end-user didn't even see what was being developed until the final user acceptance testing at the end of the project. This approach presents several opportunities for problems:

- Many users have a difficult time defining detailed requirements up front in a project, especially in a very uncertain and changing environment.
- Relying too heavily on documentation can lead to significant miscommunications and misunderstandings about the intent of the requirements.

It's important to note that this statement doesn't imply that there should be no documentation at all in an agile project. The key thing to recognize is that any documentation should provide value in some way. *Documentation should never be an end in itself*.

Customer Collaboration over Contract Negotiation

The third value statement indicates a preference for "customer collaboration over contract negotiation." The typical project prior to agile has been sometimes based on an "arm's-length" contracting approach. Project managers for many years have been measured on controlling costs and schedules, and doing that has required some form of contract to deliver something based on a defined specification. Of course, that also requires some form of change control to limit changes in the requirements as the project progresses.

Agile recognizes that, particularly in an uncertain environment, a more collaborative approach can be much more effective. Instead of having an ironclad contract to deliver something based on some predefined requirements, it is better in some cases to create a general agreement based on some high-level requirements and work out the details as the project progresses. Naturally, that approach requires a spirit of trust and partnership between the project team and the end-customer that the team will ultimately deliver what is required within a reasonable time and budget.

Again, it is important to recognize that these values are all relative, and you have to fit this statement to the situation. At one extreme, I have applied an agile approach to a government-contracting environment. Naturally, in that environment we had to have a contract that called for deliverables and milestones and expected costs, but even in that environment, the government agency understood the value of collaboration over having a rigidly defined, arm's-length kind of contract, and we were able to find the right balance. At the other extreme, you might have a project where the requirements are very uncertain and a much more adaptive approach is needed to work collaboratively with the customer to define the requirements as the project progresses, without much of a contract at all.

Responding to Change over Following a Plan

The final value statement indicates a preference for "responding to change over following a plan." This statement is in response to many projects that have been oriented toward controlling costs and schedules. They made it difficult for the customer to change the requirements in order to control the scope and, hence, the cost and schedule of the project.

The problem in applying that approach to environments where the requirements for the project are uncertain and difficult to specify is that it forces the user to totally define the requirements for a project upfront without even envisioning what the final result is going to look like, and that's just not a very realistic approach in many cases.

In many situations, it is more effective to recognize that, at some level, the requirements are going to change as the project progresses and design the project approach around that kind of change. It's important to recognize that this is not an all-or-nothing decision . . . to have either completely undefined requirements or highly detailed requirements. There are a lot of alternatives between those two extremes, and you have to choose the right approach based on the nature of the project.

All of these statements are somewhat interrelated, and you have to consider them in concert to design an approach that is appropriate for a particular project. From a project management perspective, this calls for some skill. Instead of force-fitting a project to some kind of canned and well-defined methodology like waterfall, the project manager needs to intelligently develop an approach that is well-suited to the project—that applies to any approach, not just agile.

Agile Manifesto principles

The four value statements form the foundation of the Agile Manifesto. The principles in the Agile Manifesto expand on the values and provide more detail. Again, as with the value statements, these statements should also be considered as relative preferences and not absolutes. The principles are summarized as follows and are discussed in more detail in the following sections:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a project team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.[12]

Early and Continuous Delivery of Valuable Software

> "*Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*"

The first principle emphasizes "early and continuous delivery of valuable software." In many traditional plan-driven projects prior to agile, the end-user customer doesn't see anything until the final user acceptance test phase of the project, and by that time it is very difficult and expensive to make any changes that might be needed.

Emphasizing early delivery of the software accomplishes two major goals:

1. It provides an opportunity for the customer to see the software early in the development cycle and provide feedback and input so that corrections can be made quickly and easily.
2. Working software is a good measure of progress. It's much more accurate and effective to measure progress in terms of incremental software functionality that has actually been completed, tested, and delivered to the user's satisfaction rather that attempting to measure the percentage of completion of a very large development project that is incomplete.

It is very difficult to accurately measure progress of a large software development project as a whole without breaking it up into pieces. That can be a very subjective judgment with some amount of guesswork. Breaking up the effort into well-defined pieces that each have clearly defined criteria for being considered "done" provides a much more factual and objective way of measuring progress.

[12] http://agilemanifesto.org/principles.html.

Welcome Changing Requirements

> "*Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage*."

The next principle emphasizes creating an environment where change is expected and welcomed rather than rigidly controlled and limited; but, of course, that doesn't mean that the project is totally uncontrolled. There are lots of ways to manage change effectively and collaboratively based on a partnership with the customer. The important thing is that the project team and the customer should have a mutual understanding upfront of how change will be managed.

Deliver Working Software Frequently

> "*Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale*."

The next principle emphasizes using an iterative approach to break up a project into very small increments called *sprints* or *iterations,* which are typically in the range of two to four weeks. There are a couple of reasons why this makes a lot of sense:

1. All agile development processes such as Scrum are based on continuous improvement. Instead of having a rigidly defined process that never changes, the team is expected to take an empirical approach to learn what works and what doesn't work as the project progresses, and make adjustments as necessary. If the project is broken up into very short increments and learning takes place at the end of each increment, learning and continuous improvement can happen much more rapidly. A popular agile mantra is, "Fail early, fail often." In other words, it's better in many cases to try something quickly and learn from it and make adjustments, rather than taking all the time that may be needed to try to design an approach that is going to work flawlessly the first time.
2. People work more productively given short time-boxes to get things done. If it is done correctly, the team develops a cadence and a tempo that is very efficient for producing defined increments of work quickly and efficiently, like a manufacturing assembly line.

Business People and Developers Must Work Together

> "*Business people and developers must work together daily throughout the project*."

The next principle emphasizes a partnership approach between the project team and the business sponsors. This is very consistent with the Agile Manifesto value of "collaboration over contracts."

To implement this principle, both the business sponsors and the project team need to feel joint responsibility for the successful completion of the project. This calls for a much higher level of engagement of the business sponsors than is commonly found in many traditional projects where the implementation of the project might be almost totally delegated to the project team.

The degree of engagement, of course, should be appropriate to the nature of the project and how the business sponsors get engaged might be different depending on the circumstances. For example, Scrum has a role called the *product owner* that provides the day-to-day business direction for the project, but the direction might not be limited to that. In a large, enterprise-level project, there might be a number of other stakeholders that need to provide input and need to be engaged somehow.

Designing an approach that gets the right people engaged at the right time is very important for making the project successful.

Build Projects around Highly Motivated Individuals

> "*Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done*."

The next principle emphasizes the importance of properly motivated individuals on a project. Too often in the past, some project managers have used high-pressure, command-and-control tactics to pressure project teams into delivering results faster. Many of us have been involved in "death march" projects in our careers where people are given an absolute deadline for getting something done, and have to work nights and weekends if necessary to get it done. When you're in an environment that requires high levels of creativity and innovation, that approach just doesn't work very well.

The philosophy of agile is based on a high level of empowerment and individual initiative by the people on the project. Instead of being told specifically what to do and being pressured into doing it to meet deadlines, agile teams are given general direction and are expected to figure out how to get it done most effectively and efficiently themselves. Making that kind of approach work requires a people-oriented leadership style. However, it doesn't mean that there is no need for leadership whatsoever.

An agile project manager needs to adapt his or her leadership style to fit the situation and that will typically depend on several factors including the nature of the project and the level of maturity and experience of the team.

Face-to-Face Conversation

> "*The most efficient and effective method of conveying information to and within a project team is face-to-face conversation*."

The next principle emphasizes face-to-face conversation. This is another statement that you have to not take as an absolute but think of it as relative. It is not always possible with distributed teams to have face-to-face communications, but it is certainly desirable *if it is* possible.

This statement also doesn't mean that the *only* form of communication is direct, face-to-face communications. It is a reaction to the history of waterfall projects that heavily relied on documented requirements as a way of communication. There are many ways to communicate information in various forms, and you need to choose the optimum mix to fit a given situation. The right mix will depend on a number of factors, including the scope and complexity of the project and the distribution of the team working on the project.

Working Software Is the Primary Measure of Progress

> "*Working software is the primary measure of progress.*"

Measuring progress on a software development project can be difficult and problematic. The traditional method is to break a project into tasks and track percent completion of those tasks as a way to measure progress; however, that can be very misleading, because often the list of tasks is incomplete and the level of completion often requires some subjective judgment, which is difficult to make and often inaccurate.

Testing is another factor in this—very often in the past, the entire development process and the testing process might have been sequential. The result is that even though the development of the software might have seemed to be complete, *you don't know how complete it really is until it has been tested and validated to be complete*. An agile approach emphasizes doing testing much more concurrently as the software is developed. There is a concept in agile called the *definition of done* that you will hear quite often. The team should clearly define what *done* means—it generally means that the software has been tested and accepted by the user. In other environments, the definition of *done* might be a lot more ambiguous and subject to interpretation. If you don't have a clear definition of done, any estimate of percent complete is likely to be suspect.

A more accurate measure of progress is to break up a software project into chunks of functionality where each chunk of software has a clear definition of done and can be demonstrated to the user for feedback and acceptance.

Promote Sustainable Development

> "*Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*"

Many of the underpinnings of agile come from lean manufacturing and total quality management (TQM). In a manufacturing environment, companies learned many years ago that running a

manufacturing plant like a sweatshop and forcing workers to work an excessive number of hours under poor conditions does not often result in high-quality products.

A similar thing is especially true in an agile environment, because the success of the effort is so critically dependent on the creativity and motivation of the team. In that kind of situation, it is even more important to create an environment where work is sustainable over a long period of time.

Continuous Attention to Technical Excellence and Good Design

> "*Continuous attention to technical excellence and good design enhances agility.*"

This next statement is an interesting one. Many people might have the image of an agile software project team as a bunch of "cowboys" that just get together and hammer out code without much design planning and without any coding standards. That is not usually the case.

Agile recognizes the need for doing things the right way to avoid unnecessary rework later. However, an agile approach should not result in overdesigning or "gold-plating" a product. A comment you will hear often in an agile environment is the concept of "just barely good enough." In other words, the work should be done to a sufficient level of completeness and quality to fulfill the purpose it was intended to fill, and nothing more. Going beyond that level of "just barely good enough" is considered waste.

Simplicity Is Essential

> "*Simplicity—the art of maximizing the amount of work not done—is essential.*"

This next statement emphasizes *simplicity*. How many times have we seen projects go out of control because the requirements become much too complex and very difficult to implement and the requirements become overdesigned to try to satisfy every possible need you can imagine?

This is also related to the concept of "just barely good enough"—don't overdesign something; keep it as simple as possible. In some cases, it might make sense to start with something really simple, see if it fills the need, and then expand the functionality later only if necessary. Another concept in agile is called the *minimum viable product,* which defines the minimum set of functional features a product has to have to be viable at all in the marketplace.

It's generally much more effective to take an incremental approach to start with something simple and then expand it as necessary, rather than starting with something overly complex that may be overkill for the requirement.

Self-Organizing Teams

> "*The best architectures, requirements, and designs emerge from self-organizing teams.*"

Agile is heavily based on the idea of self-organizing teams but that needs some interpretation. Sometimes, developers have used the idea of "self-organizing" as an excuse for anarchy, but that is not what was intended. The intent is that if you have the right people on a cross-functional team and the team is empowered to collectively use all the skills on the team in a collaborative manner, it will generally deliver a better result than a single individual could deliver acting alone.

Continuous Improvement

> "*At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly*."

Agile is adaptive in two respects:

- The design is adaptive to uncertain and changing user requirements—it can start with a high-level view of requirements and progressively elaborate requirements after the project is initiated.
- The process itself is adaptive rather than highly prescriptive. Agile is based heavily on continuous improvement, using short intervals to reflect on what's working and what's not working and taking quick corrective action as necessary. In Scrum, this is called a *retrospective,* and it happens at the end of each sprint.

The team is expected to continuously improve and adapt the agile process as needed as the project progresses.

SUMMARY OF KEY POINTS

1. Agile History

Agile has evolved over a number of years from a proliferation of approaches going back to the 1980s and 1990s. The Agile Manifesto which was published in 2001 was a key turning point that defined some principles and values that were needed to simplify and synthesize that diverse array of different methodologies.

2. Agile Manifest Values and Principles

The Agile Manifesto values and principles provide a strong foundation for understanding agile at a deeper level. These values and principles were not meant to be applied rigidly; they were intended to be interpreted and applied in the context of a given project and business environment.

That is the essence of an adaptive approach; rather than rigidly applying a fixed set of rules to all projects, it is much better to focus on the values and principles and use good sense to apply them intelligently to fit a given project and business environment.

Many of these values and principles are applicable to almost any project in some way. It's not a matter of having one set of values and principles for agile projects and another set of completely different values and principles for other projects; it's a matter of applying these values and principles intelligently in the right context to fit the project.

DISCUSSION TOPICS

Agile History

What do you think is most significant about the role of the Agile Manifesto in the overall history of the development of agile principles and practice up to that point?

Agile Manifesto Values

1. What are some of the trade-offs that you think might have to be made in applying the Agile Manifesto values to real-world projects? How would you go about resolving these trade-offs?
2. Provide an example of a project and discuss how you might have applied the Agile Manifesto values to that project.

Agile Manifesto Principles

3. Which two or three of the Agile Manifesto principles do you think are likely to have the biggest impact on the success or failure of a typical project, and why?
4. Are there any Agile Manifesto principles that you think would not be applicable to every project (agile or not)? Why not?
5. Provide an example of a project and discuss how you might have applied the Agile Manifesto principles to that project.

3 Scrum Overview

SCRUM HAS BECOME RAPIDLY ACCEPTED as the most widely used agile methodology in the United States and its usage is rapidly expanding in other areas of the world as well. It provides a good general foundation that can be adapted to fit a very broad range of projects. Scrum is also not limited to software development, but that is where it is most widely used at the current time. For that reason, much of the discussion in this book will be focused on software development, but it should be understood that agile and Scrum are not limited to software development. For example, I used a Scrum approach to plan and organize the writing of this book.

Scrum, is, by definition, an *empirical process* as opposed to a "defined and predictive process." The following is how these two types of processes are different:

1. *Empirical process*. The empirical process control model was defined to exercise or control the process via following some frequent adaptations as well as frequent inspections. It works best in situations with high levels of uncertainty where it is difficult, if not impossible, to clearly define the solution in advance and an experimental, trial-and-error approach is needed to converge on an acceptable solution. The term *empirical process* control model is based on the word *empirical,* which means the information is acquired by the means of experimentation and observation.[1]
2. *Defined and predictive process*. "The defined process control model can be thought of as a theoretical approach. When a well-defined set of inputs is given, it is obvious that the same outcomes will be generated every time the program executes. With the well-understood technologies and stable requirements, one can very well predict a whole software project." [2]

In other words, a defined and predictive process model is appropriate if both the requirements for the project and the solution to those requirements can be accurately predicted in advance; however,

[1] Blogspot, "Explain Empirical versus Defined and Predictive Process?" April 5, 2012, http://productdevelop.blogspot.com/2012/04/explain-empirical-vs-defined.html
[2] Ibid.

that is typically not the case in a software development project. That is why an empirical process like Scrum is typically so much more successful in any environment such as software development with high levels of uncertainty.

Scrum is adaptive in two ways:

1. *Solution adaptation*. Scrum is adaptive in the sense of progressively defining the solution. You can start a Scrum project with only a very fuzzy idea of the requirements and the solution and progressively elaborate the requirements and the solution as the project proceeds to ultimately converge on a solution that meets the business need.
2. *Process adaptation*. The process itself behind Scrum is also adaptive. A Scrum project is broken up into short, fixed-length sprints. At the end of each sprint as the project proceeds, the project team pauses to determine if the process is working or needs to be further adapted to fit the nature of the problem.

Ken Schwaber, one of the original developers of Scrum, provides this overview of Scrum:

> I offer you Scrum, a most perplexing and paradoxical process for managing complex projects. On one hand, Scrum is disarmingly simple. The process, its practices, its artifacts, and its rules are few, straightforward, and easy to learn . . . On the other hand, Scrum's simplicity can be deceptive. Scrum is not a prescriptive process; it doesn't describe what to do in every circumstance. Scrum is used for complex work in which it is impossible to predict everything that will occur. Accordingly, Scrum simply offers a framework and a set of practices that keep everything visible. This allows Scrum's practitioners to know exactly what's going on and to make on-the-spot adjustments to keep the project moving toward desired goals.
>
> Common sense is a combination of experience, training, humility, wit, and intelligence. People employing Scrum apply common sense every time they find the work is veering off the path leading to the desired results. Yet most of us are so used to prescriptive processes—those that say 'do this, then do that, and then do this'—that we have learned to disregard our common sense and instead await instructions.[3]

SCRUM ROLES

Scrum is based on some very well-defined roles that are discussed in this section.

[3]Ken Schwaber, *Agile Project Management with Scrum* (Redmond, WA: Microsoft Press, 2004), p. 1.

Product owner role

The *product owner* role in Scrum is defined as follows:

> The Product Owner is responsible for maximizing the value of the product and the work of the Development Team. How this is done will vary widely across organizations, Scrum Teams, and individuals.
>
> The Product Owner is the sole person responsible for managing the Product Backlog. Product Backlog management includes:
>
> - Clearly expressing the Product Backlog items;
> - Ordering the items in the Product Backlog to achieve goals and missions;
> - Optimizing the value of the work the Development Team performs;
> - Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next; and,
> - Ensuring the Development Team understands items in the Product Backlog to the level needed.
>
> The Product Owner may do the above work or have the Development Team do it. However, the Product Owner remains accountable.[4]

The Product Owner represents the business sponsor, is a decision maker for the project, provides direction to the team on what will be built, and prioritizes the work to be done. On a typical Scrum project, there is only one product owner; however, on large enterprise-level projects, product owners might not act alone. They might gather inputs from multiple stakeholders and, in some cases, need to coordinate with product owners of related efforts.

Relationship to Project Management Role

The product owner role has many elements of functions that might be considered project management in providing direction to the project but it goes beyond a typical project management role and includes the domain knowledge to provide the business direction to a project as well. This role is very similar to a product manager in a product development company. The product owner has some responsibilities that might be similar to a project manager, but it is a very different role. In a traditional project, a project manager doesn't define the requirements of what should be developed and that is what a product owner does.

[4] Ken Schwaber and Jeff Sutherland, "Scrum Guide" Scrum.org (July 2013), https://www.scrum.org/scrum-guide.

He has some of the attributes of a business analyst in collecting and defining requirements but the role goes well-beyond the role that a business analyst would be expected to play in being a decision maker on the product features and capabilities.

Scrum Master role

The Scrum Master role in Scrum is defined as follows:[5]

> The Scrum Master is responsible for ensuring Scrum is understood and enacted. Scrum Masters do this by ensuring that the Scrum Team adheres to Scrum theory, practices, and rules.
>
> The Scrum Master is a servant-leader for the Scrum Team. The Scrum Master helps those outside the Scrum Team understand which of their interactions with Scrum Team are helpful and which aren't. The Scrum Master helps everyone change these interactions to maximize the value created by the Scrum Team.
>
> The Scrum Master serves the Product Owner in several ways, including:
>
> - Finding techniques for effective Product Backlog management;
> - Helping the Scrum Team understand the need for clear and concise Product Backlog items;
> - Understanding product planning in an empirical environment;
> - Ensuring the Product Owner knows how to arrange the Product Backlog to maximize value;
> - Understanding and practicing agility; and,
> - Facilitating Scrum events as requested and needed.
>
> The Scrum Master serves the Development Team in several ways, including:
>
> - Coaching the Development Team in self-organization and cross-functionality;
> - Helping the Development Team to create high-value products;
> - Removing impediments to the Development Team's progress;
> - Facilitating Scrum events as requested or needed; and,

[5]Ibid.

- Coaching the Development Team in organizational environments in which Scrum is not yet fully adopted and understood.

The Scrum Master serves the organization in several ways, including:

- Leading and coaching the organization in its Scrum adoption;
- Planning Scrum implementation within the organization
- Helping employees and stakeholders understand and enact Scrum and empirical product development;
- Causing change that increases the productivity of the Scrum Team; and,
- Working with other Scrum Masters to increase the effectiveness of the application of Scrum in the organization.

The Scrum Master is what is known in agile as a *servant leader*. He/she has the responsibility to facilitate the team. He/she is not expected to provide a significant amount of direction to the team as a *project manager* would—the team is supposed to be self-organizing and empowered so the role of the Scrum Master in providing direction is limited. However, in the real world, many teams are not at a level of maturity to act effectively as a self-organizing and empowered team, and the Scrum Master might have to be somewhat of an agile coach to help the team reach that level of maturity.

Relationship to Project Management Role

Some of the responsibilities of the Scrum Master might be considered similar to a project manager. For example, the Scrum Master is expected to track and remove obstacles that are inhibiting the performance of the team. However, it is also a very different role—the Scrum Master is expected only to facilitate the team without providing much direction. There are a couple of different views on this:

- The "idealistic" view is that the team is totally empowered and self-organizing, and very little or no direction of any kind is required.
- A more pragmatic view is that not all teams are at that level of maturity, and some level of leadership is needed. The Scrum Master has to really "fill the cracks" to fit the situation.

The right solution, in my experience, has been to take an adaptive approach to provide whatever leadership is needed to help the team be successful; however, a very important key point is that this is not a traditional project management role. At the team level, there is no defined role for a project manager in a Scrum project.

Team role

The role of the team in an agile project is defined as follows:[6]

> The Development Team consists of professionals who do the work of delivering a potentially releasable increment of "Done" product at the end of each Sprint. Only members of the Development Team create the increment.
>
> Development Teams are structured and empowered by the organization to organize and manage their own work. The resulting synergy optimizes the Development Team's overall efficiency and effectiveness.
>
> Development Teams have the following characteristics:
>
> - They are self-organizing. No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into increments of potentially releasable functionality;
> - Development Teams are cross-functional, with all of the skills as a team necessary to create a product increment;
> - Scrum recognizes no titles for Development Team members other than Developer, regardless of the work being performed by the person; there are no exceptions to this rule;
> - Scrum recognizes no sub-teams in the Development Team, regardless of particular domains that need to be addressed like testing or business analysis; there are no exceptions to this rule; and,
> - Individual Development Team members may have specialized skills and areas of focus but accountability belongs to the Development Team as a whole.

The team is expected to act as a single entity where all the members of the team act collaboratively and cohesively as a single unit. A team may consist of people with different specialties. For example there could be people who specialize in development on the team and others who specialize in testing. It might also include business analysts within the team.

There are also different views regarding how a team is made up:

- The idealistic view is that everyone on the team is equally capable of performing any task required by the team.
- The more pragmatic view is that, in the real world, some specialization by skill within the team is beneficial. Having different skills within the team is not inconsistent with having good teamwork,

[6] Ibid.

in my opinion. In a sports team, you have people with very different skills and abilities who play different positions on the team that are well aligned with their unique skills and abilities. Those are very different skills, but they can still work together as a well-coordinated, cohesive team. In fact, an argument can easily be made that the team should be cross-functional and having different perspectives within the team is more effective in many circumstances than having a team of people who all think absolutely alike in all situations.

Relationship to Project Management Role

The team also has some responsibilities that might be considered similar to a project manager. Each member of the team is expected to plan their activities and be accountable for delivering the results that they committed to. In a sense, agile distributes responsibility downward into the team.

Scrum framework

Scrum is generally called a *framework* rather than a *methodology* because it is meant to provide a framework for organizing the work rather than a more specific, well-defined methodology or how the work should be done. A high-level overview of the Scrum framework is shown in Figure 3.1.

Each of the elements of the framework is discussed in more details in the following sections.

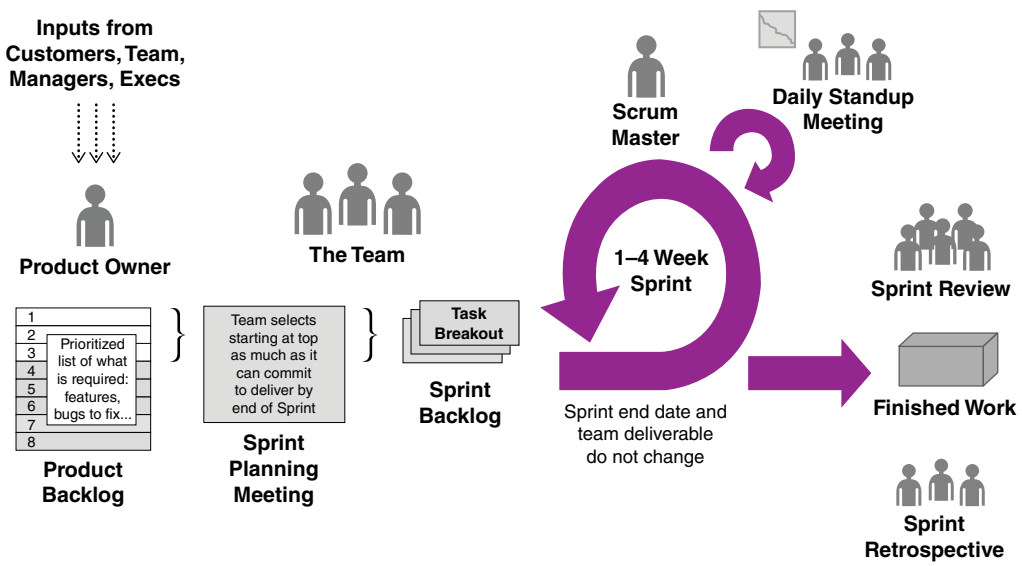


FIGURE 3.1 Scrum framework

Courtesy of Rally

Sprints

The heart of the Scrum process is the *sprint*. A "sprint is typically a fixed-length time-box and is generally from two to four weeks in duration and is where the development work is actually done. Instead of a traditional *schedule-boxing* approach in which the length of a project or an iteration is expanded as needed to be long enough for whatever is being developed, the length of a sprint is fixed and the work to be done is broken up into small increments where no single increment is so big that it cannot be accomplished in a single sprint. Instead of expanding the length of the sprint to fit the work to be done, the challenge is to see how many of these small increments of work can be fit into a single fixed-length sprint.

Product Backlog

The product backlog is a queue of the work to be done organized in the form of small increments of work typically in the form of *user stories*. (User stories will be discussed in more detail later; however, they are a very streamlined way of defining a very small and individual user requirement.) A good practice with *user stories* is to describe the user requirement in very concise terms and also briefly define what the user expects to accomplish with the user story as an acceptance criteria. The product backlog is dynamic and is continuously prioritized by the product owner to order the items in the product backlog to deliver the most business value as early as possible. As new ideas come up during the course of the project, they will also be added to the product backlog so that the product backlog continuously expands and contracts as work is completed and new work is added and is continuously reviewed, approved, and prioritized by the product owner.

The product backlog is also used to hold all work that is to be completed, including any defects that are deferred from the current sprint. Normally, defects would be resolved in the current sprint, but the product owner can make a decision to defer that work if necessary. However, building up an excessive amount of defects in the product backlog is referred to a *technical debt* and should not be allowed to grow out of control.

The product owner in Scrum is responsible for defining and managing the product backlog on an ongoing basis throughout the project; however, in many situations he/she is assisted in that role by the Scrum Master. If the project warrants it, in some cases, a business analyst may also assist the product owner with managing the product backlog.

There are different ways that the product backlog might be managed, depending on the nature of the project—in a very adaptive project approach with uncertain requirements, the product backlog might be limited to only looking two to three sprints into the future. However, that approach would not provide much of an ability to plan the schedule and costs of a project. If there is a need for longer term planning, naturally, the product backlog would need to be defined further out in time; however, it could be limited to a high-level definition only for items that are very far out in time. The product backlog will be discussed in more detail in a later section.

A very important step in the Scrum process that is not explicitly shown in this diagram is product backlog grooming. In product backlog grooming, the team (or part of the team including the product

owner) meets regularly to "groom the product backlog," in a formal or informal meeting which can lead to any of the following:[7]

- Removing user stories that no longer appear relevant
- Creating new user stories in response to newly discovered needs
- Reassessing the relative priority of stories
- Assigning estimates to stories which have yet to receive one
- Correcting estimates in light of newly discovered information
- Splitting user stories that are high priority but too coarse-grained to fit in an upcoming iteration

The product owner is responsible for product backlog grooming; the Scrum Master facilitates it; expert "leads" (SMEs, developers, QA) sometimes need to opine; and stakeholders need to contribute.

The review and refinement (especially business prioritization and sizing) of the product backlog is done in parallel *as the team is building* during a sprint. If product backlog grooming is not done, the next sprint planning will not have a good product backlog ready for the product owner and team to agree on what the next sprint needs to include. Instead, they may have to spend a day or two doing the evaluation, which could delay the start of the next sprint.

Sprint planning

Sprint planning plays a very important role in Scrum. The sprint planning meeting takes place prior to the beginning of every sprint. It has two major goals and is typically broken into two parts to align with those two goals:

1. The goal of the first part is for the product owner and the team to negotiate what stories will be taken into the sprint.
   - The product owner makes any necessary final revisions in the priority ranking of the potential stories to be taken into the sprint.
   - The team decides how many of those stories can be completed in the sprint based on the level of effort required for each story and the estimated team capacity or velocity.

Note

> "Velocity is the measure of the throughput of an agile team per iteration. Since a user story, or story, represents something of value to the customer, velocity is actually the rate at which a team delivers value. More specifically, it is the number of estimated units (typically in story points) a team delivers in an iteration of a given length and in accordance with a given definition of 'done.'"[8]

[7] "Backlog Grooming," Agile Alliance, http://guide.agilealliance.org/guide/backlog-grooming.html.
[8] "Agile Sherpa–Velocity," VersionOne, http://www.agilesherpa.org/agile_coach/metrics/velocity/.

This part of the planning meeting is essentially a negotiation between the product owner and the team.

2. The goal of the second part is for the team to define the tasks that will be needed to implement those stories and to plan how those stories and tasks will be allocated among the members of the team. The product owner typically does not participate in this portion of the sprint planning meeting.

   The product backlog should be groomed and prioritized by the product owner prior to the sprint planning session; however, final adjustments in the priority ranking are likely to take place in the sprint planning meeting as the team and the product owner negotiate and plan what can be taken into the sprint and resolve any trade-offs.

The sprint planning meeting is typically time-boxed to four hours. The largest part of that time is typically spent in the second part of task-level planning by the team. During the sprint planning meeting, the product owner and the team should agree on an overall goal to be accomplished in the sprint in addition to negotiating the stories that will be taken into the sprint. Once that is done, the team typically does the task-level planning portion of the meeting without the product owner.

Daily standup

The Daily Standup is basically a check-in for everyone on the team to coordinate what's going on, monitor progress, and to identify any obstacles that may be inhibiting progress. During the Daily Standup, each person on the team typically answers three questions:

1. What did you accomplish yesterday?
2. What are you going to accomplish today?
3. What obstacles are in your way?

The meeting is facilitated by the Scrum Master and it is often done in front of a Scrum board that indicates the progress of the tasks in the current sprint. Figure 3.2 shows an example of a Scrum board. Alternatively, for distributed teams, an online agile project management tool is typically used in lieu of a physical Scrum board. (See Chapter 10 VersionOne Tool Overview.)

Usually, everyone on the team stands up during these meetings as an incentive to keep the meeting short and focused (that's why it's called a *Standup*). However, it is common to deviate from those rules with distributed teams. With distributed teams, the Daily Standup may be one of only a few opportunities for communication among the team, so it might be necessary to go beyond the three basic questions, but it is a good idea to still limit the length of the meeting and defer lengthy topics to other meetings.

Sprint review

The sprint review is where the team presents the finished work to the product owner for his/her final review and approval. It is essentially a brief User Acceptance Test at the end of each sprint. It should

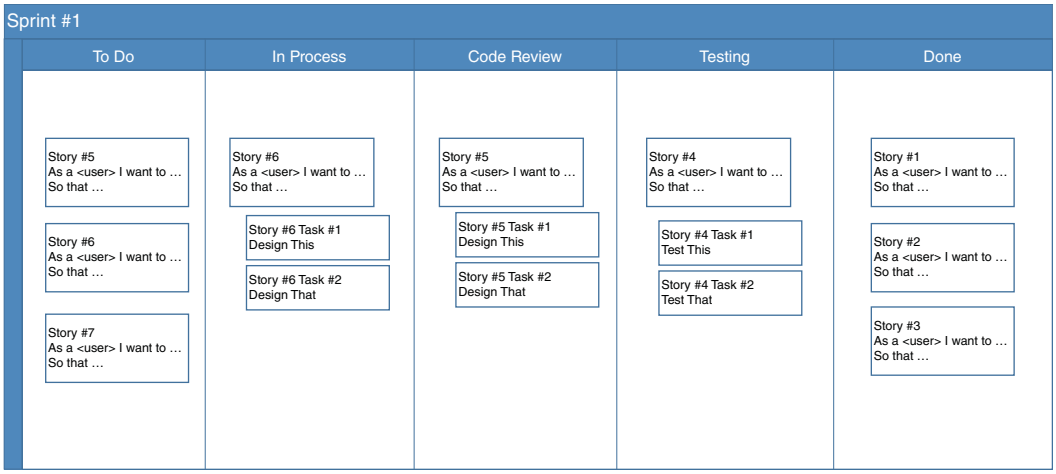


FIGURE 3.2 Example Scrum board

be noted that the sprint review should not be the very first time that the product owner has seen the results of the software that the team is developing. The team should demo and preview the results of their work to the product owner as it is being developed during the sprint to get feedback and inputs. It is also a good practice to not wait until the sprint review for final acceptance of each story. It's a good idea to present a story to the product owner for review as soon as it has been completed in the sprint. The sprint review is essentially a formal review of all the items in the sprint.

All defects in the product should typically be resolved prior to the sprint review unless the product owner has specifically approved deferring the resolution to a later sprint. If the product owner requests significant changes or enhancements in the stories, those will typically be added to the product backlog and deferred until a future sprint. Significant changes to stories that are in progress should not be allowed in the middle of a sprint—changes while the sprint is in progress should be limited to clarification and refinement of user stories only.

Although the product owner has primary responsibility for approving the results of the sprint review and should represent the interests of all business users, it is often a very good practice to include business users and other stakeholders who might have input into the sprint review in these meetings.

Sprint retrospective

The sprint retrospective is an opportunity to review and discuss lessons learned at the end of the sprint where the team looks back, reflects on what went well and what didn't go well, and identifies opportunities for process improvement in the next sprint. This is a very important element of the Scrum process. A sprint retrospective is to a sprint in an agile project as a post mortem is to a project

in a traditional project. It is an opportunity to stop and reflect on the process and make improvements. Because it is done at the end of each sprint and sprints take place every two to four weeks, learning and process improvement can be much more rapid than a traditional project. An agile development process is not intended to be rigid; it is intended to rely heavily on continuous improvement to adapt the process to the project and the environment as much as possible. Because sprints are short, learning can be very fast so that mistakes and problems are not allowed to propagate very far.

GENERAL SCRUM/AGILE PRINCIPLES

It's important to not get lost in the mechanics of Scrum and to understand the principles behind it in order to apply the methodology intelligently in a variety of different project and business environments.

> People tend to interpret Scrum within the context of their current project management methodologies. They apply Scrum rules and practices without fully understanding the underlying principles of self-organization, emergence, and visibility and the inspection/adaptation cycle. They don't understand that Scrum involves a paradigm shift from control to empowerment, from contracts to collaboration, and from documentation to code.[9]

Kenneth Rubin has provided a very nice summary of the important principles behind Scrum in his book, Essential Scrum: A Practical Guide to the Most Popular Agile Process[10] that are further discussed here. These principles are really not unique to Scrum; they are general enough to apply, to some extent, to almost any project management approach (either plan-driven or adaptive). They just have to be used intelligently in the right proportions based on the nature of the project.

Variability and uncertainty

An understanding of variability and uncertainty and how to manage them effectively is probably one of the most critical requirements for any agile project manager to master. Kenneth Rubin identifies several important principles associated with managing variability and uncertainty:[11]

- *Embrace helpful variability*. Plan-driven processes are built around controlling and limiting change. In a plan-driven process, any change is considered an exception to the norm and must be controlled in order to manage the scope, costs, and schedule of the project. That can lead to a very

[9]Schwaber, *Agile Project Management with Scrum,* p. 26.
[10]Kenneth Rubin, *Essential Scrum: A Practical Guide to the Most Popular Agile Process* (Upper Saddle River, NJ: Pearson Education, 2013).
[11]Ibid., p. 32.

inflexible process where you may wind up meeting cost and schedule goals but fail to provide the desired business value because the process wasn't adaptive enough to meet customer needs. This is probably the primary reason agile has been so successful in a very dynamic and rapidly changing business environment.

An agile process is based around easily adapting to customer needs and embracing change; however, change in an agile process is not totally unrestricted and a sensible approach is still needed to manage changes.

- *Employ iterative and incremental development*. Agile is heavily based on an incremental development; instead of trying to build the entire product all at once, the product is broken up into smaller pieces and built incrementally with an opportunity for feedback and learning after each increment is completed.
- *Leverage variability through inspection, adaptation, and transparency*. Scrum is heavily based on inspection and adaptation. Not only is the product being built continuously and inspected and adapted as needed to maximize the value it provides to the user, but the process itself used to build the product is also continuously inspected to determine if the process is working optimally and any adjustments are made to the process as necessary. Transparency about both the product and how the process works is essential to accomplish that.
- *Reduce all forms of uncertainty simultaneously*. In any project there are a variety of kinds of uncertainty:
  - End uncertainty deals with uncertainty associated with the features of the final product.
  - Means uncertainty surrounds the process and technologies used to develop the product.
  - Customer uncertainty relates to who the ultimate customer of the product will be and what their needs are.

  A plan-driven project attempts to remove all uncertainty about the product and how it will be built up front, and that's just not realistic in many situations. An agile project is built around recognizing, managing, and reducing uncertainty as the project progresses.

Prediction and adaptation

Many people have tried to implement Scrum dogmatically and rigidly "by the book," when it was intended to be adapted to the nature of the project. An important set of principles behind Scrum identified by Kenneth Rubin is related to prediction and adaptation. The following are the key elements of this principle that he has identified:[12]

- *Keep options open*. Plan-driven approaches frequently try to make all decisions about the requirements upfront to resolve any uncertainty about the project before it starts. Scrum and agile are

[12] Ibid., pp. 37–39.

based on delaying decisions until "the last responsible moment". That is, in general, you should delay making decisions as long as they can be delayed without impacting the project. The reasoning behind that is that if you delay making decisions, you will typically have better information for making those decisions and ultimately make better decisions. If you try to make decisions too far in advance it will frequently require speculation and many times that speculation will be wrong and the effort may be wasted because that decision will only need to be re-planned later when better information is available.

- *Accept that you can't get it right up front*. An important principle in Scrum and agile is to accept that some amount of trial-and-error experimentation may be necessary to find the best solution. This requires a mature attitude to recognize that we "don't know what we don't know."
- *Favor an adaptive, exploratory approach*. There is an economic decision associated with weighing the cost of experimenting and adapting to the results of the experimentation versus the cost of designing the product up front and the risk of major rework and redesign if it is wrong. In the 1990s, the cost of making changes was significant, which is one of the reasons that led to developing heavily plan-driven approaches. However, tools and technologies have gotten a lot better since that time, and the cost of exploratory development has come down significantly.
- *Embrace change in an economically sensible way*. Agile and Scrum provide sensible ways to manage change to minimize the economic impact to the project of those changes. With traditional plan-driven approaches, the cost of changes can rise significantly later in the project because the impact can be significant. By using an incremental approach to development in an agile project, the impact of changes can be limited and the costs of making changes can be relatively flat throughout the project.
- Balanc*e predictive up-front work with adaptive just-in-time work*. In any project, there is always a certain amount of information that is known or can be easily known upfront and some that is much more difficult to determine upfront. It would be foolish to ignore what is already known. A sensible approach is to balance a predictive up-front approach with an adaptive just-in-time approach to resolve uncertainties based on the nature of the project.

Validated learning

A third area of principles that Kenneth Rubin has identified is related to "Validated Learning."[13] As we've mentioned, agile is based heavily on an empirical and experimental approach to try things and see what works. To make that approach work, it is important to recognize that we don't have all the answers and we might have to make some assumptions, but it is important to recognize that those assumptions are only assumptions and need to be validated.

[13] Ibid., pp. 44–46.

- *Validate important assumptions fast*. If there are specific assumptions or decisions that might have a significant impact on a number of areas of the project that might require significant rework if they're not made correctly, it makes good sense to identify those areas and resolve them as quickly as possible to minimize that impact.
- *Leverage multiple concurrent learning loops*. There are multiple concurrent learning loops in a Scrum project that operate at different levels, and it is best to take advantage of all of these different levels concurrently. For example, there is a level of learning that takes place in the daily Scrum meetings and there is also a level of learning that takes place at the end of each sprint in the sprint review and sprint retrospective. We should take advantage of all these sources of learning.
- *Organize workflow for fast feedback*. We should organize the workflow in the project so that we get feedback as rapidly as possible on the items where feedback is most essential to resolve uncertainties and validate assumptions.

Work in progress

The next group of principles identified by Kenneth Rubin is related to work in progress, or WIP. The important elements of the principles that he has identified in this area are:[14]

- *Use economically sensible batch sizes*. Traditional, plan-driven development processes have been based on the principle of economy of scale that comes from a manufacturing environment. In a manufacturing environment, it may make sense to perform a repetitive manufacturing operation on large batch sizes to reduce costs, but there are very different economies of scale associated with a product development process and small batch sizes tend to be more efficient in that environment for a number of reasons:
  - Work can be distributed more evenly and flow through the process is maximized where large batch sizes can cause serious bottlenecks and reduce overall flow.
  - Faster cycle time means feedback is quicker, which results in faster learning and adaptation.
  - Risk is also reduced.
- *Recognize inventory and manage it for good flow*. Manufacturing plants are very much aware of the cost and impact of carrying a large amount of inventory. In a software development process, "inventory" is not as visible, and the impact of carrying inventory is not as well understood. Inventory in a product development process consists of a variety of different kinds of items of work in process including requirements, code, and so on. There is always a cost and a certain amount of overhead associated with managing those items of inventory, so it is best to keep work in process inventory as small as possible to make the overall process more efficient.

[14] Ibid., pp. 48–52.

- *Focus on idle work, not idle workers*. There are two kinds of waste that are important to recognize in a project:
  - One is *idle worker waste* where workers are not being fully utilized at 100 percent of their capacity and some component of their capacity is not being used and wasted that might be better utilized.
  - The other is *idle work waste*, which is when work sits idle until people are available to work on it.

    Both of these forms of waste are important, but traditional, plan-driven projects are typically very heavily focused on *idle worker waste* and attempt to make sure that everyone in the project is working at 100 percent of capacity. An agile project recognizes that both of these forms of waste are important and *idle work waste* is at least equally important if you want to optimize the flow of work through the process.
- *Consider cost of delay*. A related consideration is the cost of delay. If a project is delayed waiting for resources to be available, the impact on the cost of the project might be significant because everyone on the project might be delayed for some period of time. The example that Kenneth Rubin gives is that delaying the completion of documentation until the end of the project might delay the entire project by two to three months, and there could be a significant cost in that delay.

Progress

Another set of principles identified by Kenneth Rubin is related to progress. He has identified the elements of these principles as follows:[15]

- *Adapt to real-time information and replan*. Traditional projects are based on conformance to plan, which overlooks the fact that the plan could be wrong or the plan might need to be adapted to changing conditions as the project progresses. An agile/Scrum project is much more adaptive and is based on the assumption that the plan will be continuously revised based on learning and information that emerges throughout the process.
- *Measure progress by validating working assets*. Traditional projects often measure progress by estimating percent complete of the tasks required to complete the project. That is typically based on a very subjective judgment that can be very inaccurate.

  A traditional project also has some major risks to progress that aren't fully considered in evaluating progress. For example, a task might be complete but if it isn't tested and accepted by the customer, is that really complete? A much better way to measure progress is measuring completed working assets that have been validated and accepted by the customer as they are being built.

[15] Ibid., pp. 54–55.

- *Focus on value-centric delivery*. Traditional, plan-driven projects typically perform final integration, testing, and delivery of all features at the end of the project. With this approach, there is a risk that the project will run out of time and money before delivering all of the important value to the customer. Also, we all know that there are many traditional, plan-driven projects that are bloated and/or "gold-plated" where a large percentage of the features are not really essential. This often results from the customer's belief that if they don't get some item that they might want into the requirements up front, they may never get it at all.

  Agile works by prioritizing the features to be delivered based on value and developing and by delivering those features incrementally, we can avoid the "all or nothing" approach that typically occurs in traditional, plan-driven projects and at least deliver the most important value to the customer quickly. There have been many situations where after some percent of the value is delivered, the customer believes that his essential needs are satisfied, there may be diminishing returns associated with completing the rest of the functionality, and there may be no need to complete the remainder of the project. This can result in a considerable savings in development costs.

Performance

The final set of principles identified by Kenneth Rubin is related to performance. He has identified the elements of these principles as follows:[16]

- *Go fast but never hurry*. An agile project is based on being nimble, adaptive, and speedy; however, rushing too quickly to get things done may not be the best approach and may have undesirable side effects such as burning out the people who work on the project and perhaps even compromising the quality of the product.
- *Build in quality*. In a traditional, plan-driven project, testing might typically be done in a later phase so the approach to quality might be somewhat reactive and focused on finding and fixing defects well after the development has been completed. There are a number of very significant potential problems associated with that approach:
  - Deferring finding problems until very late in the project, where it might be much more difficult to resolve the problems, can make detecting and resolving individual defects much more difficult. When problems are allowed to accumulate without being resolved, there is a compounding effect that can make it very difficult to isolate and resolve individual problems. It's like trying to untangle a gigantic ball of twine with lots of knots in it.

    As an example, I worked at a large electronics company in the early 1990s that spent over $150 million developing a large, complex communications switching system. Much of the testing to integrate all of the components was deferred until the very end, and because it was such

[16] Ibid., pp. 57–58.

a complex system, it became very difficult to isolate individual problems. The whole project was canceled, and the company had to default on delivery commitments to several very large and important beta test customers.

- There is also a large risk that problems won't be found at all if testing is not directly associated with incremental functionality as it is developed and testing is delegated to an independent group that doesn't have primary responsibility for developing the product. In that situation, the people doing the testing may not be fully aware of the functionality that has been developed; and, as a result, the testing may be incomplete.

  Agile is heavily based on the principles in TQM, which emphasizes a more proactive process to eliminate defects at the source rather than finding and fixing them later. Manufacturing environments learned that lesson a long time ago. Years ago, there used to be a lot of quality control inspectors at the end of an assembly line that would inspect products and reject any that were defective. That was a costly process, not only in terms of products that had to be discarded or reworked but also in the cost of the many inspectors it took to find those defects. Also, any inspection approach like that is based on sampling, and it would be prohibitively expensive to inspect 100 percent of the product so some defects are bound to get through undetected.

  Going upstream in the process and designing the process to be inherently reliable in producing products that were free of defects, companies were able to significantly reduce the cost of QC inspectors and produce much more reliable products. Also, putting the responsibility on the people doing the development of the product to produce quality products that are free of defects rather than relying on an independent group to try to find defects later in the process has a very significant impact.

- *Employ minimally sufficient ceremony*. Traditional, plan-driven projects tend to be high ceremony, document-centric, process-heavy approaches. Scrum and agile use a much leaner process where ceremony, documentation, and process are limited as much as possible to only items that contribute value to the project.

  A good example is documentation. There is a common misconception that an agile project does not produce any documentation. That is not the case but documentation should not be an end-in-itself. We shouldn't produce documentation for the sake of documentation. If a document serves an important purpose and provides value in some way, it should be included in the project, but there are many ways to simplify documentation to make it satisfy the purpose it was intended for more efficiently.

  Here's an example of a streamlined approach to documentation: instead of writing detailed functional specs to define a set of features, some simple user stories can be created, and some very succinct acceptance tests can be defined in conjunction with the user stories to better define the requirements that story must satisfy. Those very simple user stories and very succinct acceptance criteria can eliminate the need for highly detailed functional specifications and, in most situations, provide a more effective solution.

SCRUM VALUES

Having a consistent set of values among everyone who participates in a Scrum team plays a critical role in helping to develop a collaborative, cross-functional approach. When people act from common values, it helps to unify everyone on the team, reduces potential conflict, and provides a solid foundation for guiding the team. The following is a summary of the Scrum values.

Commitment and focus

The first Scrum value is commitment and focus. As stated in the Scrum Alliance Code of Ethics:

> Commitment is our willingness to dedicate ourselves to a goal and to do our best to meet that goal. Focus means that we concentrate on and are answerable for doing the things that we have committed ourselves to do, rather than allowing ourselves to become distracted or diverted.
>
> As practitioners in the global Scrum community:
>
> - We take responsibility for and fulfill the commitments that we undertake—we do what we say we will do.
> - We make decisions and take actions based on the best interests of society, public safety, and the environment.
> - When we make errors or omissions, we take ownership and make corrections promptly. When we discover errors or omissions caused by others, we promptly communicate them to the appropriate individual or body. We accept accountability for any issues resulting from our errors or omissions and any resulting consequences.
> - We protect proprietary or confidential information that has been entrusted to us.
> - We proactively and fully disclose any real or potential conflicts of interest to the appropriate parties.[17]

Commitment is very important:

› The team needs to agree on a goal and the items that will be accomplished for each sprint during the sprint planning meeting and the team should hold itself accountable for successfully completing those items and goals as the sprint progresses.

[17] *Scrum Alliance Code of Ethics,* http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB0QFjAA&url=http%3A%2F%2Fmembers.scrumalliance.org%2Fresource_download%2F1687&ei=tJTmU5OBL8-dygSO_YLADQ&usg=AFQjCNFUdwoEFFD_gqPoN4g2j7hzANCqdw&sig2=GdjGuSvPhIf_tYCLOsmXGg&bvm=bv.72676100,d.aWw.

› The idea of team accountability is a very important notion in Scrum. The whole idea of a self-organizing team relieves a lot of burden on the need for someone to manage the efforts of the team. If it works properly, there is no need for a project manager to provide direction to the team and track progress at the team level.

The idea of focus is also very important—once the items to be completed during the sprint have been agreed to, they should not be radically changed in the middle of the sprint. It is understood that details associated with items will be elaborated during the sprint; however, nothing should be allowed that changes the focus of the team during the sprint.

Openness

The second Scrum value is openness. Being able to be open with others is a sign of emotional maturity and it is essential for developing high performance teams that work collaboratively with each other. We've also seen immature people and teams who have hidden agendas and political goals that might supersede and conflict with the goals of the team.

Once people have worked on Scrum teams and have learned to practice these values, it becomes much easier to assemble a team who has shared values, but it can be difficult to get an immature team with no prior experience in Scrum to this level. The *Scrum Alliance Code of Ethics* says that as practitioners in the global community:

- We earnestly seek to understand the truth.
- We strive to create an environment in which others feel safe to tell the truth.
- We are truthful in our communications and in our conduct.
- We demonstrate transparency in our decision-making process.
- We provide accurate information in a highly visible and timely manner.
- We make commitments and promises, implied or explicit, in good faith.
- We do not engage in or condone behavior that is designed to deceive others.[18]

Teams typically go through stages of maturity in developing openness with each other. An excellent model of team dynamics is the *Tuckman model* that describes several different stages of team dynamics:[19]

1. Forming
    - No prior experience working with others on the team; team members may be somewhat reserved, formal and polite with each other'

[18] Ibid., p. 2.
[19] "Forming, Storming, Norming, and Performing," Mind Tools, http://www.mindtools.com/pages/article/newLDR_86.htm.

- Some members of the team may be apprehensive and unsure of themselves with the rest of the team.
- There is uncertainty about how the team will evolve.

2. Storming
   - People on the team begin to openly challenge each other.
   - Some conflict occurs and is essential to get past before the team can progress.
3. Norming
   - The team agrees to work together around some defined rules.
4. Performing
   - The team reaches the highest level of performance and works naturally and collaboratively and is able to dynamically adjust to new situations.

The theory is that teams need to go through these stages in order to make progress, but it isn't necessarily totally sequential. Sometimes teams will move back and forth between stages in order to make progress. For example, *storming* and conflict should be viewed as a sign of progress because it is essential for teams to go through that stage before making real progress towards the higher level of "performing" as a truly cross-functional and collaborative team based on openness.

Respect

Showing respect is a very important Scrum value. It is well known that high-performance teams may be composed of people with different opinions. If everyone on the team was a yes man and went along with everything others on the team said automatically, the team would probably not be very high performing. There is a lot of value in hearing different points of view and reaching consensus, and that calls for respect. The *Scrum Alliance Code of Ethics* states it this way:

> Respect means that we show a high regard for ourselves, others, and the resources entrusted to us. Resources entrusted to us may include people, money, reputation, the safety of others, and natural or environmental resources.
>
> An environment of respect engenders trust, confidence, and performance excellence by fostering mutual cooperation and collaboration—an environment where diverse perspectives and views are encouraged and valued.
>
> As practitioners in the global Scrum community:
>
> - We respect the rights and beliefs of others.
> - We listen to others' points of view, seeking to understand them.
> - We approach directly those persons with whom we have a conflict or disagreement.
> - We conduct ourselves in a professional manner, even when it is not reciprocated.

- We negotiate in good faith.
- We do not exercise the power of our expertise or position to influence inappropriately the decisions or actions of others in order to benefit personally at their expense.
- We do not discriminate against others based on, but not limited to, gender, race, age, religion, disability, nationality, or sexual orientation.
- We do not engage in any illegal behavior.[20]

Courage

The final Scrum value is courage. In my experience, this value becomes important in several different ways:

1. Scrum is difficult and challenging, and doing it successfully requires courage to face and overcome many obstacles.
2. It requires facing up to reality openly and honestly.
3. You need courage to overcome your own personal inhibitions to interact with others. The *Scrum Alliance Code of Ethics* states it this way:

> Courage means that we have the daring to do the best that we can and the endurance not to give up. We have the determination and resolution to take ownership of the decisions we make or fail to make, the actions we take or fail to take, and the consequences that result.
>
> As practitioners in the global Scrum community:
>
> - We share bad news even when it may be poorly received.
> - We avoid burying information or shifting blame to others when outcomes are negative.
> - We avoid taking credit for the achievements of others when outcomes are positive.
> - We would rather say, "No," than make false promises.
> - We accept the possibility of failure but also know that we can learn from failure and apply those learnings to our next attempt.
> - We acknowledge that change is inevitable, that change leads to growth, and that growth guides us toward improvement.
> - We admit when we need help and we ask for help.[21]

[20] *Scrum Alliance Code of Ethics*, p. 2.
[21] Ibid., p. 3.

SUMMARY OF KEY POINTS

1. Scrum Overview

   Scrum is based on an empirical process control model; it is adaptive at two levels (1) the definition of the solution is adaptive and (2) the process itself is adaptive.

2. Scrum Framework

   Many people think of agile as being very loosely defined, with a very minimally defined process—that is not really accurate. The Scrum process is actually very well defined and requires a good deal of skill and discipline. It is a different kind of discipline in that the process itself is intended to be very dynamic and adaptive.

3. General Scrum/Agile Principles

   It's very important to not get lost in the mechanics of how to do Scrum—having a defined methodology is important, but it is intended as a basis for ongoing continuous improvement. Both the process and the project deliverables are expected to evolve with the project based on rapid learning at the end of each sprint.

   It's also very important to understand the principles behind Scrum in order to do whatever adaptation may be necessary to apply it to unique projects and business environments.

4. Scrum Values

   Values in Scrum are important. Having a consistent set of values among everyone who participates in a Scrum team plays a critical role in helping to develop a collaborative, cross-functional approach. When people act from a common sense of values, it helps to unify everyone on the team, reduces potential conflict, and provides a solid foundation for guiding the team.

DISCUSSION TOPICS

Scrum Overview

1. Explain the difference between an empirical process control model and a defined and predictive process control model. When would each make sense?

Scrum Roles

2. How do you think the overall need for project management changes in a Scrum project, and how would the functions of a traditional project manager be absorbed into the Scrum roles on a typical Scrum team?

3. What are some of the most important differences between Scrum roles and traditional project roles, and why is the shift in roles important in an agile environment?

Scrum Methodology

4. What is the role of the product backlog in a Scrum project? How does it differ from the requirements definition practices in a traditional project?

5. When is the sprint planning meeting held? What are the major parts of a sprint planning meeting? What is the output of the sprint planning meeting?
6. What is the purpose of the sprint review meeting, and when is it held? What is the desired output of the sprint review meeting?
7. What is the objective of the sprint retrospective meeting, and how is it different from a sprint review meeting?
8. Who attends the daily standup meeting, and what is its purpose?

General Scrum/Agile Principles

9. Which general Scrum/agile principle do you think might have had the greatest impact on projects you have been involved in, and why?
10. Discuss a project and how you might have used the general Scrum/agile principles to do the project differently. Why?
11. Which general Scrum/agile principles might be the most difficult to implement, and why?

Scrum Values

12. Why are values important? How does having a consistent set of values affect the performance of a Scrum team?
13. Which Scrum value is likely to have the most impact on team performance? Why?
14. If a team is having conflict, what might that indicate? What action would be appropriate to resolve the conflict?

4 Agile Planning, Requirements, and Product Backlog

AGILE PLANNING PRACTICES

PROBABLY THE BIGGEST DIFFERENCE between agile and more traditional project management approaches is in the area of planning. There's a misconception that agile projects do not require planning. That is not the case—they require just as much or more planning; it is just done very differently.

Rolling-wave planning

Traditional, plan-driven project management approaches typically attempt to do more of the planning prior to the start of the project, while agile project management approaches typically defers planning decisions to "the last responsible moment." By *the last responsible moment*, we mean the latest point in time that a decision can be made without impacting the outcome of the overall project.

That is what is called *rolling wave planning*:

- You typically start with a high-level plan that is sufficient for defining at least the vision, scope, and objectives of the project to whatever level of detail is needed at that point to support whatever level of planning and estimation is required for the project.
- The details of the plan and the requirements are further elaborated as the project progresses.

The thinking behind that strategy is that attempting to plan too far in advance naturally involves some amount of guesswork and speculation. Quite often, that speculation is wrong and will result in wasted effort in replanning later, and it might also require reworking any work that has been done based on erroneous assumptions. Deferring planning decisions as long as possible should result in better decisions because more information will be available at that point in time to make those decisions with less speculation.

Rolling-wave planning is not really new to agile. The general concept has been around for a long time and has been in PMBOK® for some time. However, in actual practice, the plan-driven or waterfall approach typically requires doing more of the planning up front. Prior to agile, iterative approaches have recognized the need for taking a more incremental approach to planning and building a solution. Iterative approaches also deferred detailed planning of each iteration until when that level of detail is needed.

Planning strategies

Planning is very much related to management of uncertainty in a project, and the planning approach generally attempts to remove uncertainty by more clearly defining the goals and requirements for the project. A plan-driven or waterfall project attempts to reduce the level of uncertainty associated with the project to a very low level before the project starts, as shown in Figure 4.1. That might be a reasonable approach with some projects, but it's not reasonable to force that kind of approach on a project that has very high levels of uncertainty that can't be easily reduced.

Attempting to do that on a project with very high levels of uncertainty forces you to make a number of assumptions about the requirements for the project based on very sketchy and incomplete information, and the quality of those assumptions may be very questionable. Sometimes, project managers make assumptions like that, lock them into a requirements document to define the project, and then make it difficult to change those assumptions through a formal method of change control later. That creates an *illusion of control*; on the surface, it looks like the project is very well controlled,

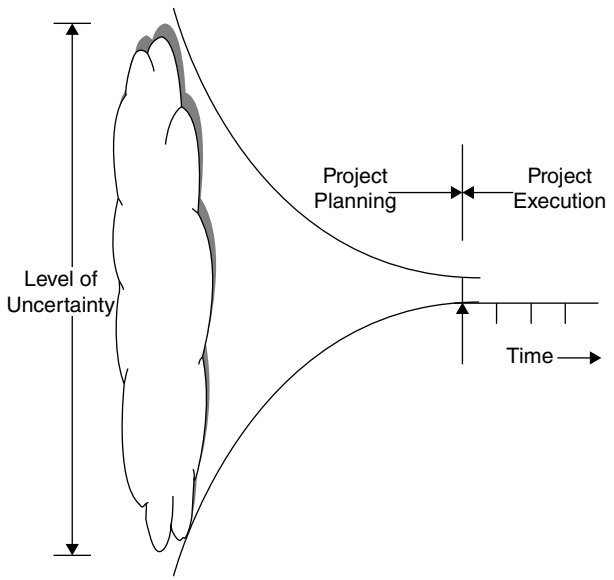


FIGURE 4.1 Typical plan-driven or waterfall planning approach

but a project can only be as well controlled as the requirements are certain. As a result, you might start the project with what appears to be a very detailed plan that is designed to control the project, and after some large number of change requests later, discover that it wasn't so well-controlled after all. An agile approach recognizes that and doesn't attempt to resolve all of the uncertainty associated with a project prior to the start of the project.

A key point I want to make is that the level of planning in an agile project can be very different, depending on the nature of the project. It's a difference between taking a highly adaptive approach and a totally plan-driven approach. The approach should be adapted to fit the nature of the project. There are a number of factors that influence which approach is most appropriate for a given project, but probably the biggest factor is the level of uncertainty in the project.

For example, if you were building a bridge across a river, it would be ridiculous to say something like, "We're going to build the first span, see how that comes out, and then we'll decide how to build the remaining spans." A typical bridge-building project would be very plan-driven and all the requirements, as well as the design of the bridge, would typically be well-defined to a fair amount of detail prior to beginning to build the bridge. On the other hand, what if there was some uncertainty in the project? Suppose that the river called for building a new kind of bridge that had never been built before, or there was some uncertainty about how stable the foundation for the bridge is? You might want to prototype and test the new design on a smaller scale somehow to remove some of the uncertainty and risk before committing to the full-scale development.

The key things to remember:

- You should always fit the methodology to the project based on the characteristics of the project and other factors such as the training and capabilities of the project team
- One of the biggest factors that influences the selection of an approach is the level of uncertainty in the project.
- It is not a black-and-white decision to be totally plan-driven or totally adaptive. There are plenty of alternatives between those two extremes.

Figure 4.2 shows the general way that an agile rolling-wave planning approach progressively reduces the uncertainty in a project as the project progresses rather than attempting to remove all the uncertainty up front. Breaking up the project into releases and iterations means that a lot of the detailed planning can be deferred until that release or iteration is ready for development (as I've mentioned, the approach need not be this sophisticated).

Spikes

Many times, there is a lot of uncertainty associated with requirements and/or what the most likely solution to those requirements should be. It's a good technique to isolate and resolve this uncertainty separately from the development effort. Letting too much uncertainty get into the development process can seriously undermine the efficiency of the team.

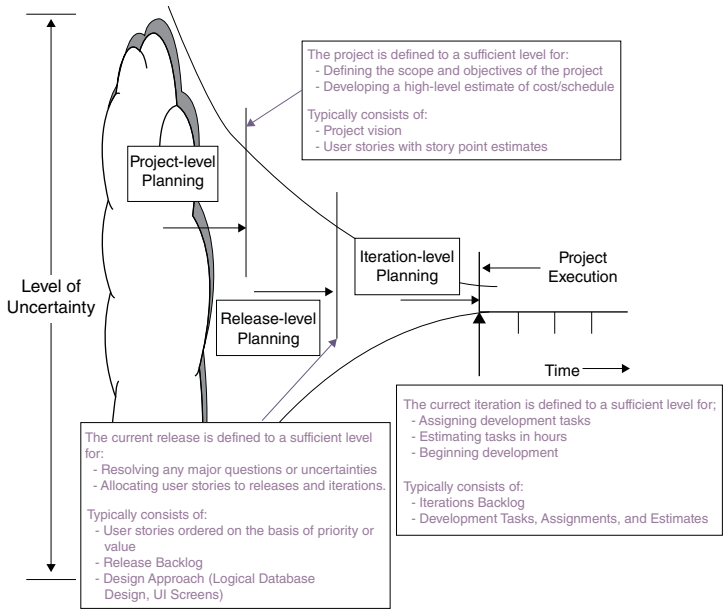


FIGURE 4.2 Agile planning approach

For that reason, a technique called a *spike* is often used in an agile project to resolve uncertainty. A spike is a special kind of iteration that is used to do research, possibly prototype a solution, and/or evaluate alternative approaches for resolving uncertainty associated with developing a solution.

Isolating and time-boxing the amount of time dedicated to resolving this kind of uncertainty can help prevent it from significantly slowing down the mainstream development effort.

Progressive elaboration

The concept of *rolling-wave planning* is very much related to the concept of *progressive elaboration,* which is an important concept in planning a strategy for the definition of requirements in an agile project. However, it is not really new to agile. Progressive elaboration has been a traditional project management practice for a long time, but agile emphasizes its use much more heavily.

The key point is that you shouldn't get too bogged down in planning requirements too far in advance because it involves too much speculation and may lead to unnecessary rework if that speculation is wrong. Requirements should only be defined to the extent needed to support whatever decisions or action is required at that particular point in time. For example:

- At the project-level, there may be a need to understand the overall scope of the project to evaluate the resources, costs, and schedule required; however, that can typically be done on the basis of a very high-level understanding of the requirements without details.
- At the release-level, there may be a need for a little more accuracy in the estimated time and effort to complete a release. That might call for understanding the requirements in a little more detail, but that also should not require a significant amount of detail.

Finally, at the sprint level, there are typically two needs that should be satisfied prior to the start of a sprint:

- The team needs to have a reasonable estimate of the level of effort associated with the requirements to determine if they can fit into the capacity for a sprint.
- The team should not take any requirements into a sprint that have major uncertainties or issues associated with them that might block or delay the development effort.

However, even at the sprint level, many details of how the requirements will be implemented can be worked out while the sprint is in progress without any significant impact on the sprint-level estimates.

Value-based functional decomposition

An important benefit of an agile approach is increased focus on business value. A good approach is to start with a vision statement that clearly defines the business value that the solution will provide. A vision statement should be short and succinct. This is an example of a format that can be used for a vision statement to keep it simple:

- For (target customer)
- Who (statement of the need or opportunity)
- The (product name) is a (product category)
- That (key benefit, compelling reason to buy)
- Unlike (primary competitive alternative)
- Our product (statement of primary differentiation)

Once a high-level vision statement has been defined, it is a good technique to use functional decomposition to break down that vision statement into the functionality that will be needed to achieve that overall vision. Functional decomposition becomes particularly important on large projects where there could be hundreds of user stories. It provides a hierarchical approach for organizing requirements, which can be an essential technique for prioritizing requirements. Without an effective approach for functional decomposition that aligns with an overall value statement, it is very easy to "get lost in the weeds" of individual requirements or user stories and lose sight of the big picture of the value you're trying to deliver.

AGILE REQUIREMENTS PRACTICES

The role of a business analyst in an agile project

A business analyst (BA) many times plays a key role in a traditional project in working with the users to analyze and document requirements prior to development. However, since agile emphasizes direct

face-to-face communications between the project team and the users as much as possible with less emphasis on requirements that are documented in detail, there is less of a need for a business analyst and the role, if any, is very different.

In an ideal agile environment, there is no role for a business analyst; the product owner is directly responsible for defining and communicating the requirements to the developers in the form of user stories. However, in the real world that is not always possible:

- For large projects, the workload on the product owner could be very large.
- For complex projects, additional analysis work may be needed to further analyze requirements and to evaluate alternative approaches.

The important points are that:

- The business analyst should not become too much of an intermediary, isolating the project team from the product owner and the business users and inhibiting communications.
- The business analyst should play a supporting role to enable and facilitate more effective communications with the project team.

Frequently, the business analyst will facilitate discussions between the developers, the product owner, and the business users to perform product backlog grooming and to further elaborate and define requirements as the project progresses.

In summary, if a business analyst is used in an agile project, the primary functions are to support the product owner by doing the following:

- *Analyze a broadly defined area and use functional decomposition to define high-level epics and stories to create a well-organized, value-driven framework to provide the required business value in the product backlog*. If the stories follow a logical functional hierarchy, it provides a mechanism for better understanding the relationship of the stories and epics to each other and for satisfying overall business goals.
- *Write individual stories that are very clear and concise and are easy to understand and implement by the project team*. Writing effective user stories is a skill that is often taken for granted. What is often overlooked in good stories is the "why" or the "so that" clause that expresses the business value the story is intended to provide. A good BA can provide that perspective that is difficult for a developer to provide.
- *Identify related user stories and epics, grouping them into a logical structure as necessary and documenting the interrelationship and associated business process flows as necessary*. The interrelationship of user stories and epics should be well-understood and the implementation of stories across different functional areas may require some planning and coordination so that they are consistently implemented. This overall framework can provide a mechanism for easily identifying any inconsistencies and/or missing functionality.

- *Integrate the needs of various stakeholders to produce an overall solution*. This is more crucial on large projects or programs, where there might be a need to integrate the needs of related projects as well as the needs of a number of different stakeholders.

"Just barely good enough"

A common problem in many traditional projects is that requirements can become bloated. One of the big reasons that happens is that users feel that if they don't ask for everything that they could possibly need and get it into the requirements, they might not get it at all. Also, we all know that in many situations in the past, project teams have gone beyond the basic requirements and "gold-plated" a solution. Excellence was perceived as going above and beyond the user requirements to develop a more complete solution, but going far beyond the user needs to develop a solution isn't necessarily a good thing and can significantly increase the scope and complexity of the development effort.

It may seem to be counter-intuitive to traditional project management thinking, but a very important principle in agile is simplicity and limiting a solution to what is "just barely good enough" to solve the problem. There is an optimum point, and beyond that point, adding additional features has diminishing value. It requires close collaboration with the users to find where that optimum point is. A good technique is to start with the simplest and most basic solution possible and then add to it incrementally and iteratively *only* to the extent that it adds value to the user.

An important element in making this work is a spirit of trust and partnership between the users and the project team. The users have to trust that if they don't ask for something specific and it is discovered to be important as the project progresses that it can be easily added at that point if necessary.

Differentiating wants from needs and the "five whys"

Another very good technique in developing requirements for agile projects is to differentiate wants from needs.

- Wants tend to be associated with a solution that a client envisions.
- Needs tend to be associated with the problem.

It is typical in many instances for a user to tell you what they think they *want* for a solution before the problem the solution is intended to solve is clearly defined. You should never lose sight of the problem to be solved before you go too far into implementing a solution. It is also a good practice to dig a little deeper into the problem to be solved to get to the root cause of the problem to be solved.

The *five whys* method is a good approach for digging into a customer need to get to the root of the problem. The idea is that by progressively asking "why" over-and-over again, you eventually get to the root cause of the problem. The following shows an example of how the "Five Why's" technique can be used to get to the root of a problem:[1]

[1] "5 Whys: Getting to the Root of a Problem Quickly," MindTools.com, http://www.mindtools.com/pages/article/newTMC_5W.htm

1. *Why is our client, Hinson Corp., unhappy?* Because we did not deliver our services when we said we would.
2. *Why were we unable to meet the agreed-upon timeline or schedule for delivery?* The job took much longer than we thought it would.
3. *Why did it take so much longer?* Because we underestimated the complexity of the job.
4. *Why did we underestimate the complexity of the job?* Because we made a quick estimate of the time needed to complete it, and did not list the individual stages needed to complete the project.
5. *Why didn't we do this?* Because we were running behind on other projects. We clearly need to review our time estimation and specification procedures.

MoSCoW technique

Another very good technique for prioritizing requirements is called *MoSCoW,* which is defined as follows:[2]

- *Must have*—Requirements labeled as "*MUST*" have to be included in the current delivery time box in order for it to be a success. If even one "*MUST*" requirement is not included, the project delivery should be considered a failure.
- *Should have*—"*SHOULD*" requirements are also critical to the success of the project, but are not necessary for delivery in the current delivery time box.
- *Could have*—Requirements labeled as "*COULD*" are less critical and often seen as *nice to have*.
- *Won't have*—"*WON'T*" requirements are either the least-critical, lowest-payback items, or not appropriate at that time.

An example of how this approach has been used successfully is the case study of General Dynamics, UK, which is discussed in Chapter 21 of this book. In that situation, General Dynamics, UK, was faced with the challenge of managing a large fixed-price government contract, and the only way to do it successfully was to work with the government customer to prioritize the requirements and eliminate some of the requirements that weren't absolutely essential to the solution using the MoSCoW method.

USER PERSONAS AND STORIES

User personas

A *user persona* is used in agile for personalizing user requirements; however, it really is a general process that can be used in any kind of development effort. The idea behind a *user persona* is that many

[2]MoSCoW Method, http://dsdm.org/content/10-moscow-prioritisation

traditional requirements management practices are impersonal; they result in a large documentation with lots of specific requirements that the system must meet, but it is not clear who those requirements are designed to satisfy. Agile puts a strong emphasis on providing value to the user and also has a strong emphasis on engaging the user directly in the project to provide feedback and inputs as the project progresses. Organizing the project requirements around specific users and their needs puts more focus on really understanding the value that the project provides and *who* the recipient of that value is.

To facilitate that process, it is useful to identify *user personas* as specifically as possible so that the project team can target their development efforts at the needs of those specific users. A user persona could be a specific category of user or it could even be a specific user, but in either case, it is useful to model that user's personality and specific interests as a hypothetical user persona. A user persona helps the team visualize that user and focus its efforts around the user's needs.

The following is an example of a user persona:[3]

User: Fred Fish, Director of Food Services "Get me out of the office and into the kitchen"

Background: Fred is director of foodservices for Boise Controls, a mid-sized manufacturer of electronic devices used in home security systems. He uses a computer, but he's a chef by trade and not so *computer-savvy*.

Key Goals: As a manager, Fred doesn't get his hands (literally) dirty the way he used to. He stops in at all the Boise Controls sites and sticks his fingers into things once in a while to stay in touch with cooks and cooking. He wants to learn computer tools but not at the expense of managing his kitchens. A computer is just another tool for getting his administrative tasks done.

A Usage Scenario: At the start of every quarter, he meets with the head chefs and plans out the next quarter's menus. That's one of his favorite things, because each chef gets to demonstrate a new meal. The chefs spend time in the kitchen exploring each new dish. When they're done, he sends the food to his staff and his manager.

He's not a computer whiz. On a good day, he can drag in some clip art and do some formatting with fonts. Once in a while, he'll format menus with the new editor on his MacBook Pro.

User stories

User stories are a succinct way of defining requirements in agile. Telling user stories is a way of simplifying the definition of the requirements in a language that can be easily understood by both developers and users. It breaks the requirements into small chunks of functionality that can be built incrementally. User stories follow the general format shown below:

As a <role> I want <to be able to do something> so that <benefit>

[3]A Sample Persona, http://creativebeatle.files.wordpress.com/2010/12/sample-persona-from-interaction-design.pdf

A user story consists of some standard parts:

- The first part identifies *who* the actor (or role) is that needs to do something. This should be as specific as possible to clearly identify who the story needs to satisfy.
- The next part identifies succinctly *what* the user (role) needs to do so that the end result is very clear.
- The third part is the "so that" clause. This clause provides some insight into *why* this functionality is needed. Knowing this additional insight should help the developers get a deeper understanding of the user's need. (What business value does it provide?)

Here are two examples:

As a student I want to purchase my monthly parking passes online so that I can save time.

As a student, I want to be able to pay for my parking passes via a credit card to avoid using cash.

There are several key advantages of a user story:

- It provides a standardized and concise way of defining a requirement in simple English that is easy for everyone to understand (both developers and users).
- It encourages defining requirements in small, bite-sized chunks where the functionality to be developed is clearly defined and can be completed within one sprint.

An important aspect of user stories is that they are written in functional terms—they define an expected result and don't tell the developer *how* to design the software to achieve that result.

- User stories are intentionally designed to be brief and don't typically provide a detailed description of what is required. They are intended to be a "placeholder for a conversation."
- That detail should take place through face-to-face communications; however, many times acceptance criteria are included in user stories to more clearly define the expected result.

There's a mnemonic that is well-known in agile called "INVEST" that is a useful way of defining the essential characteristics of good user stories.[4]

Independent: Stories should be as "independent" as possible so that they can be worked on in any order. That will simplify the flow of stories through development and avoid bottlenecks that can be caused by having too many dependencies among stories.

Negotiable: Stories should be "negotiable"—a story is a placeholder for conversation, and some dialog is expected to take place to explore trade-offs associated with developing the story as efficiently and as effectively as possible.

[4]Bob Hartman, "New to Agile? INVEST in Good User Stories," Agile for All, May 14, 2009, http://www.Agileforall.com/2009/05/14/new-to-Agile-invest-in-good-user-stories/.

Valuable: Stories should be "valuable"—the value that a story is intended to produce should be clearly-defined so that the product owner can make an objective evaluation of the level of effort required versus the value to be gained from the story.

Estimable—"Estimable" is the next important characteristic. A story needs to be sufficiently defined so that the team can develop a high-level estimate of the effort required for the story in story points.

Small—Stories should be relatively "small" so that functionality can be developed and tested as incrementally as possible. Breaking the work into small chunks allows it to flow much more smoothly and allows the work to be distributed more evenly among the team while large efforts can easily lead to bottlenecks and inefficiencies in distributing work among the team.

Testable: Finally, stories should be "testable" to determine if they have successfully fulfilled the value proposition that they are intended to fulfill. It's a good practice with agile stories to write acceptance test criteria, along with the stories to define the tests that they need to fulfill. The test criteria essentially take the place of more detailed specifications for what the story must do.

Epics

An *epic* is basically a very large user story. An epic serves the purpose of associating related individual user stories with a higher-level purpose that they are collectively intended to fulfill, but an epic is normally too large for the project team to work on directly without breaking it down into individual user stories.

It's a useful technique on large, complex projects for organizing user stories into some kind of structure so that the interrelationship of user stories is well-understood. The following shows an example of a large epic and how it can be broken down into smaller stories.

Epic:

> *As an electronic banking customer, I want to be able to easily make an online deposit of a check into my bank account through my IPhone® so that I can save the time required to send a check for deposit through the mail and I can have the money immediately credited to my checking account as soon as the deposit is completed electronically.*

Stories:

- As an electronic banking customer, I want to be able to scan an image of the front and back of a check into the IPhone® so that it can be deposited electronically.
- As an electronic banking customer, I want to be able to enter the deposit information associated with an electronic deposit so that the correct amount will be deposited into the correct bank account when the electronic deposit is processed.

- As an electronic banking customer, I want to be able to electronically submit a scanned check and deposit information to the bank for deposit so that I can save the time associated with sending deposits by mail.
- As an electronic banking customer, I want to be able to receive confirmation of a completed electronic deposit so that I will know that the deposit was successfully processed.

PRODUCT BACKLOG

What is a product backlog?

The product backlog was previously discussed in Chapter 3. It typically consists of user stories, and it is dynamic. The user stories are continuously groomed and prioritized over the course of a project. It is essentially a queue of work to be done. There are different ways to implement a product backlog and there are different schools of thought on how it should be done.

At one extreme, there is a view that the project should be totally adaptive and you shouldn't try to define the product backlog any more than two to three sprints into the future. That approach might work fine for some projects, but a problem with that approach is that it doesn't provide a basis for planning the overall scope, costs, and schedule of a project. If there are some expectations about the cost and schedule of the project, you may have to define the product backlog more completely, at least at a high level to evaluate the scope of the project to support those estimates. However, the ability to do that is obviously related to the level of uncertainty in the project.

You should develop an approach for planning the requirements and managing the scope of the project that is appropriate for the project. I would say that the most significant factors in selecting an approach are:

1. The level of uncertainty in the project and the difficulty of defining the requirements for the project upfront, and
2. The need for estimating some kind of schedule and costs for the project.

Figure 4.3 shows the flow of information into the product backlog. Product owners are responsible for defining and maintaining the product backlog; however, they might be assisted in that effort by the Scrum Master or others.

Product backlog grooming

There is a very general approach for thinking about how to organize the product backlog. Think of it as an "iceberg" in which the items in the iceberg gradually make their way to the top as they become ready for development, as shown in Figure 4.4.

The principles of *rolling-wave planning* are used and items start out at the bottom of the backlog as being very roughly defined and the stories are progressively "groomed" as they move closer

FIGURE 4.3 Product backlog flow

Courtesy of Rally

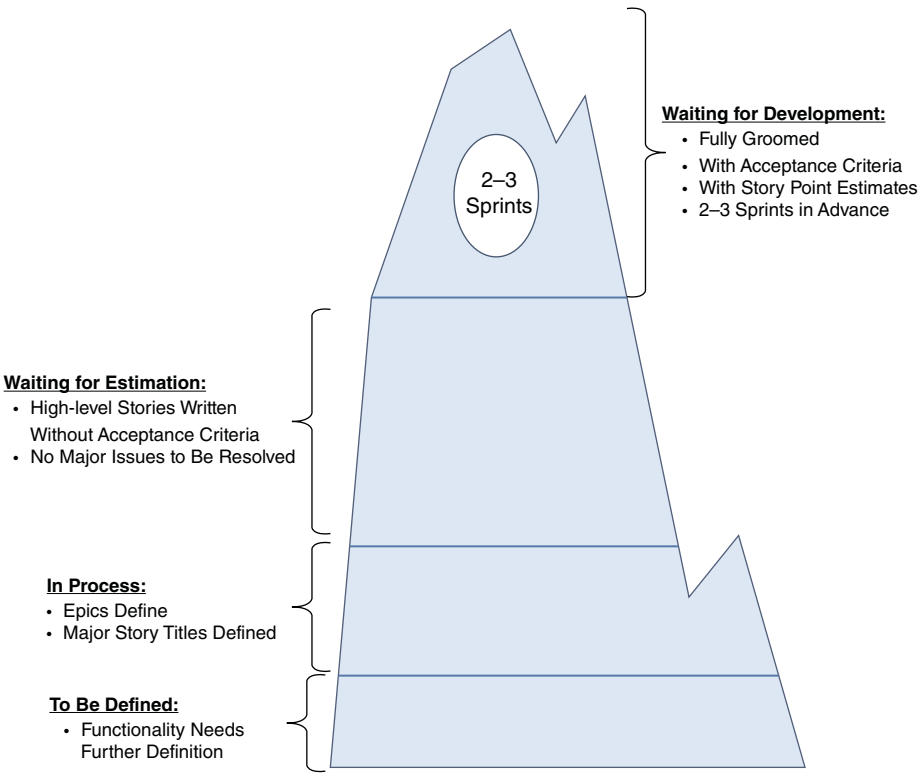


FIGURE 4.4 Product backlog grooming flow

to development. Grooming is an important part of any agile project but it is often overlooked and not planned for as much as it should be. A good technique is for the project team to allocate some time in each sprint so that the queue of stories to be developed never runs empty.

Agile is based heavily on "just-in-time" planning—obviously, you never want a project team waiting around for stories to be developed so you have to manage the grooming process to get stories ready for development as needed, but on the other hand, you don't want to work any farther in advance than necessary, because some of that effort might only need to be repeated once the stories are closer to being ready for development.

It is perfectly reasonable to develop a high-level backlog far enough out in time to develop an estimate of the scope, costs, and schedule of the project. It is a judgment call of how far it is reasonable to try to plan into the future. Some will say that it's a waste of time to plan any more than two to three sprints in advance, but if you have a need to estimate the costs and schedule of a project, you have to define the product backlog further out in time at a sufficient level to support those estimates and then elaborate the details later.

SUMMARY OF KEY POINTS

1. Agile Planning Practices

There is a common misconception that agile projects are not planned at all. That is not the case, it is just a different kind of planning, and the planning is spread out over the project rather than attempting to do the majority of the planning upfront prior to the start of the project. Rolling-wave planning and progressive elaboration of requirements are both used to plan agile projects.

The planning approach is very much related to the level of uncertainty in a project and the planning approach attempts to reduce the level of uncertainty to an acceptable level based on the nature and complexity of the project. The level of planning in an agile project can vary significantly depending on the project and it is always best to fit the planning approach to the nature of the project and the level of uncertainty associated with it.

2. Agile Requirements Practices

There is no defined role for a business analyst in an agile project as there is in a plan-driven or waterfall project and the product owner is ultimately responsible for defining and prioritizing the project requirements; however, the product owner might be assisted in that role by a business analyst. If a business analyst is engaged in an agile project, the BA should not become too much of an intermediary and isolate the project team from the product owner and users and inhibit communications—the BA should play a supporting role and facilitate more effective communications with the project team.

Prioritization of requirements is extremely important in an agile project and requires a spirit of trust and collaboration between the business users and the project team to work together to deliver

the highest-value items first and to preserve simplicity of the solution and avoid going beyond what is needed to satisfy the fundamental business need.

3. User Personas and Stories

   A user persona is used in agile for personalizing user requirements; however, it really is a general process that can be used in any kind of development effort. User stories are a succinct way of defining requirements in agile. Telling user stories simplifies the definition of the requirements in a language that can be easily understood by both developers and users. It breaks the requirements into small chunks of functionality that can be built incrementally. User stories follow the general format: *As a <role> I want <to be able to do something> so that <benefit>*. An epic is a large user story.

4. Product Backlog

   The product backlog is a very important tool in an agile project for maintaining a prioritized queue of project requirements. It should be dynamic and should be continuously groomed as the project progresses. Agile projects generally use an iceberg strategy for grooming the product backlog. The items that are near the top of the iceberg and are closest to going into development should get the most attention. There should typically be about two to three sprints worth of stories at the top of the backlog that are well-groomed and ready to go into development in order to avoid a situation where the project team is waiting for work to do. However, the product backlog grooming does not need to be limited to two to three sprints in advance, and it is a judgment call of how far it is reasonable to plan into the future.

DISCUSSION TOPICS

Agile Planning Practices

1. What is rolling-wave planning? How is it different from a traditional project to an agile project? Why would it make sense to use rolling-wave planning?
2. What are the levels of planning that you might typically find in an agile project? What is the role of each?
3. What is progressive elaboration, and why does it make sense?
4. What is a spike, and how is it used? Provide an example.

Agile Requirements Practices

5. What is the role of a business analyst in an agile project? When would a BA typically be needed and why?
6. Explain the concept of value-based functional decomposition and how it would be used. Provide an example of a real world project and how that concept would be used in the project.

7. What is the "five why's" technique, and how would it be used? Provide an example of how it might be used from a real world project.
8. What is the MoSCoW technique, and how is it used? Provide an example of how it might be used.

User Personas and Stories

9. What is a user persona? Give an example of a user persona in a project. How does it help to define the requirements?
10. What are the advantages of using user stories to define requirements in an agile project? Write an example user story for a project that you have been engaged in.
11. What are the characteristics of well-written user stories? Provide an example of a user story that is not well written and that violates these characteristics.

Product Backlog

12. What purpose does the product backlog serve? What are the characteristics of a well-organized product backlog?
13. Who is responsible for the product backlog in an agile/Scrum project?

5 Agile Development, Quality, and Testing Practices

YOU MIGHT ASK: "Why does a project manager need to know something about development practices?" In the past, the role of a project manager might have been somewhat limited to a coordination function to integrate the efforts of different functional organizations that played a role in the project. In many cases, the actual direction for the different functions involved (development, test, etc.) came from the managers who were responsible for those functions themselves, and the role of the project manager in providing direction may have been limited. An agile environment is different:

- Instead of a relatively loosely knit team of people from a variety of functional departments who might only work on a particular project on a part-time basis, an agile team is typically dedicated to a project and should be much more tightly integrated.
- The methodology for an agile project is much more of one well-integrated project methodology with all the functions (development, test, etc.) working more collaboratively and concurrently. Any decisions about how development and testing is done need to be integrated with the overall project management approach.

Those factors require a much higher level of cross-functional leadership. Agile project managers can help provide that leadership if they have the cross-functional knowledge that is required, but it is more than a simple coordination function.

AGILE SOFTWARE DEVELOPMENT PRACTICES

This section provides an overview of some of the most important agile software development practices that an agile project manager should be familiar with.

Code refactoring

In order to improve the reliability and maintainability of the software, code refactoring involves:

- Removing redundancy
- Eliminating unused functionality
- Rejuvenating obsolete designs
- Improving the design of existing software

Agile approaches tend to emphasize quickly creating code to meet functional requirements *first* and then refactoring the code as necessary *later* to clean it up. Refactoring throughout the entire project life cycle saves time and increases the quality of the software.

In a traditional project, there is typically an emphasis on laying out the entire design and architecture of the project early in the project to avoid unnecessary rework later. In an agile project, that approach may not be very realistic, for a couple of reasons:

1. It's very difficult in an agile approach to define all the requirements in a sufficient level of detail at the beginning of the project to lay out the entire design up front.
2. An agile approach encourages changes as the project progresses in order to better adapt to user needs and requirements and attempting to define and control the design approach early in the project may restrict that ability to make changes.

The question then becomes—how do you avoid having code that is a complete mess and is unreliable and difficult to maintain as a result of so many unplanned changes? Fortunately, modern development tools have made it easier to implement code refactoring as the project progresses to more easily rework and clean up the code to make it more maintainable and reliable in an agile environment. The use of design patterns and coding standards can also make this approach more practical to implement.

The truth is that code refactoring is just as important in a traditional project, but it often hasn't been done adequately because it was assumed that the design of the code was defined and stabilized up front, and that just isn't a very realistic assumption in many cases, even in a non–agile environment. The strengths and benefits of a code refactoring approach are that it:

Here are the strengths and benefits of code refactoring:

- Encourages developers to put the primary focus on the functionality provided by the code first and clean it up later to make the code more well-structured and maintainable.
- Reduces the time required to produce functional code that can be available for prototyping and user validation.

There are some risks and limitations associated with code refactoring:

- The amount of rework of the code required might be significant.
- The structure of the code might not be optimized around the most desirable architectural approach.

- Time pressures to complete the iteration may short-circuit the code-refactoring effort and allow poorly organized code to be released.

Continuous integration

Continuous integration is the practice of frequently integrating new or changed software with the code repository and performing overall system integration testing throughout the project rather than deferring that effort until the end of the project. Continuous integration provides a way of early detection of problems that may occur when individual software developers are working on code changes that might conflict with each other. In many typical software development environments, integration might not be performed until the application is ready for final release.

I was involved in a large software development project in the early 1990s where a major electronics company invested over $150 million in the development of a very complex hardware/software switching system and the project had to be abandoned in the end because when it was time to integrate the project, the various components of the system could not be integrated and the overall system would not work reliably. That is a perfect example of the critical importance of continuous integration.

There are some strengths and benefits of continuous integration:

- Developers detect and fix integration problems continuously.
- Early warning of broken/incompatible code and of conflicting changes.
- Constant availability of a "current" build for testing, demo, or release purposes.
- Immediate feedback to developers on the quality, functionality, or system-wide impact of code they are writing.
- Frequent code check-in pushes developers to create modular, less-complex code.
- Metrics generated from automated testing and continuous integration focus developers on developing functional, quality code, and help develop momentum in a team.

There are also some risks and limitations associated with continuous integration:

- It might be difficult to implement on larger code development projects where the integration effort may be too complex to do as frequently.
- It requires a level of sophistication and close teamwork on the part of the team to make it work especially on large, complex projects.
- Resources and tools to automate the continuous integration, building, and testing process are essential and can be expensive.

Pair programming

Pair programming is another agile development process that is sometimes used to improve the quality and reliability of software. Pair programming is analogous to the way a pilot and a copilot fly a

commercial airliner. In a commercial airliner, one of the pilots flies the airplane while the other pilot oversees the overall flying of the aircraft and provides support to the person who is actually doing the flying. Pair programming in a software development environment works in a similar fashion.

One developer typically writes the code and the other developer provides overall guidance and direction as well as support and possibly mentoring to the person writing the code. This technique might be used by two peer-level developers who rotate between the two roles, or it might be used by a senior-level developer to mentor a more junior-level developer. Pair programming is not as widely used as some other agile development practices because the economics of dedicating two programmers to work together on the same task can't always be justified. In some cases, code reviews are substituted for pair programming to achieve a similar effect.

Here are the strengths and benefits of pair programming:

- Design quality:
  - One developer observing the other person's work should result in better quality software with better designs and fewer bugs.
  - Any defects should also be caught much earlier in the development process as the code is being developed.
  - The cost of a second developer that is required may or may not be at least partially offset by productivity gains.
- Learning and training. Sharing knowledge about the system as the development progresses increases learning.
- Overcoming difficult problems. Pairs are able to more easily resolve difficult problems.

There are also some risks and limitations associated with pair programming:

- Work Preference
  - Some developers prefer to work alone.
  - A less-experienced, less-confident developer may feel intimidated when pairing with a more experienced developer and might participate less as a result.
  - Experienced developers may find it tedious to tutor a less-experienced developer.
- Costs
  - The productivity gains may not offset the additional costs of adding a second developer.

Test-driven development

Test-driven development is a somewhat widely used agile development practice. It is typically used in conjunction with unit testing by developers. Rather than writing some code first and then, as a second step, writing tests to validate the code, the developer actually starts by writing a test that the code

needs to pass to demonstrate that it does the functionality that was intended and then writes code to make that test succeed. Of course, the test fails initially until the functionality has been implemented, and the objective is to do just enough coding to make the test pass. Development is done incrementally in very small steps—one test and a small bit of corresponding functional code at a time.

Here are the strengths and benefits of test-driven development:

- It encourages the development of small, incremental modules of code that can be easily tested and integrated with a continuous integration process
- It is well suited to testing of the design as it progresses and provides immediate feedback to the developers on how well the design meets the requirements
- It also encourages developers to write only the minimal amount of code necessary to pass a given test

There are also risks and limitations associated with test-driven development:

- Test-driven development primarily addresses only unit testing of code modules—much more testing is typically needed at different levels of the application.
- Test-driven development emphasizes rapidly implementing software to provide a minimum level of required functionality and relies on later refactoring the code to clean it up to meet acceptable design standards. If that refactoring isn't done, the code might not be reliable and supportable.

Extreme programming (XP)

At one time, extreme programming (commonly referred to as XP) was a more important agile methodology, but XP is primarily just a collection of agile development practices consisting of:

- User stories
- Release planning
- Iteration planning
- Test-driven development
- Collective ownership of code
- Pair programming
- Continuous integration
- Ongoing process improvement

It doesn't provide a management framework for how an agile project should be managed, as Scrum does; it is more of a development process. Scrum does provide that management framework and has become much more dominant and more widely used agile process. Sometimes XP might be

used to define the development process used with Scrum, but more often, people take more of an à la carte approach to selecting the development processes that they use with Scrum.

AGILE QUALITY MANAGEMENT PRACTICES

For the same reason that cross-functional agile project managers need to understand development practices, they also need to understand quality management practices.

Key differences in agile quality management practices

Table 5.1 shows some of the key differences between the agile approach to quality management and the approach typically used for quality management in more traditional projects.

Definition of "done"

A very important concept in agile is the definition of "done." In agile, the goal of each sprint is to produce a "potentially shippable" product. Of course, *potentially shippable* can mean different things to different people, depending on the nature of the project. In a large, enterprise-level project, it might be impossible or impractical to really release something to production at the end of each sprint, for a lot of reasons. For example, the results might not be sufficiently complete to deliver a complete subset of functionality that is required, or additional integration work might be needed to integrate the software with other related applications before it is fully released to production.

TABLE 5.1 Key Differences between Agile and Traditional Quality Management

	Typical Traditional Plan-Driven/ Waterfall Quality Management Process	Typical Agile Quality Management Process
Integration of Testing with Development	Testing is typically done sequentially with development and is done by a separate organization.	Testing is done more concurrently with development and is well-integrated into the development team.
Testing Approach	There is a more reactive approach to finding and correcting defects.	There is a more proactive approach to preventing defects.
Responsibility for Quality	Responsibility for the quality of the software is perceived as on the QA organization.	The overall team has responsibility for the quality of the software.
Regression Testing	Because testing is deferred until the end of the project and code has stabilized, there may not be a need for a complicated regression testing approach.	Because testing is done concurrently with development and the code is constantly changing, it is more essential to do regression testing as the project progresses.

The important thing is that the team should have a clearly defined definition of what "done" means so that it is unambiguous and well understood by everyone on the team. One of the biggest difficulties on a software project is that there could be different interpretations of what "done" means.

- Is it when the coding is complete?
- Is it when the coding and testing are complete?
- Does it depend on user acceptance?

The more clearly and crisply this is defined removes a lot of subjectivity of making an assessment of whether something is "done" or not and makes it clear what the team is committing to when it makes a commitment to complete some number of stories in a sprint.

The definition of "done" typically includes:

- Code review
- Tested by developers and QA
- Accepted by the product owner and other stakeholders if necessary

The important thing is that the team takes complete responsibility for the quality of the software it develops and defines its own standards of completeness—quality isn't someone else's responsibility.

The role of QA testing in an agile project

That leads right into the question of: "What is the role of QA in an agile project?" As with many things in an agile environment, there is not necessarily a single right/wrong way of doing things, and there are different schools of thought on this:

- There's a somewhat idealistic view that everyone on the team should be capable of doing anything required (development or testing), and all developers should be totally responsible for the quality of the work that they produce rather than relying on someone else on the team to test it and provide feedback.
- In actual practice, that view is very difficult to implement, and there is value in having people with focused skills within a project team. There is also value in having an independent tester look at software other than the person who developed it.

Some people would say that having people focused on specific tasks and skills within a team inhibits teamwork. I don't agree with that point of view. The analogy I like to use is of an athletic team—on a high-performance athletic team you have people who are highly trained and skilled around playing particular positions on the team, but that doesn't inhibit them working together as a cohesive team.

A good team should be cross-functional and collaborative, and having people on the team representing different perspectives will ultimately strengthen the team. QA is a discipline that needs focus

and training to do it well. To give it to a developer as just another thing for them to do might not be the best use of resources, and there's always value in having an independent observer look at someone else's software. A trained and specialized QA tester will typically see things that the primary developer might have taken for granted or overlooked.

The important points are that:

- The team, as a whole, owns responsibility for quality—*it is not someone else's responsibility*.
- QA should be an integral part of the team, not a separate organization external to the team.
- QA testing is an important skill, and it is often worthwhile to have people on the team who are specialized and skilled in QA testing who are dedicated to planning and executing QA tests.

AGILE TESTING PRACTICES

Concurrent testing

The biggest difference in agile testing practices is that testing is integrated with development in each sprint in a very collaborative process where testers and developers take joint responsibility for the quality of the product that they produce. In actual practice, there are a number of ways that can be done. For example, instead of waiting until the end of the sprint to turn over all stories to QA for testing, a better practice would be to have the testers begin testing the software as soon as it is sufficiently complete for testing. That approach allows more concurrency of development and testing and avoids any last-minute surprises at the end of the sprint.

Acceptance test driven development

Acceptance test-driven development is an excellent approach that is used in agile projects. It is very similar to test-driven development but it is at a higher level of functionality. Test-driven development is done at the level of unit testing to validate the implementation of code, while acceptance test-driven development is done at a higher level of functional testing and tests the features and behaviors of the system that are observable by the user.

Acceptance test-driven development involves writing the acceptance tests for the functionality that must be provided in each story prior to starting development. This is usually done as part of the writing or grooming of the stories, and it helps to build a more concise, common understanding of exactly what needs to be done to satisfy the user need for each story.

A big advantage of acceptance test-driven development is that it can eliminate the requirement for writing detailed functional specifications, it simplifies the requirements definition process, and it keeps everyone on the team clearly focused on the functionality that must be provided to satisfy the business need.

Repeatable tests and automated regression testing

Because testing is done concurrently with development in an agile project and new functionality is being continuously added, it is essential for tests to be repeatable in an agile project to ensure that new functionality or other development activity has not broken something that was already tested and completed. As any item of new functionality is added, a regression test is needed to repeat the testing of any items of functionality that were previously tested to ensure that nothing has been inadvertently broken by adding the new functionality.

As agile projects get larger, it becomes impractical and perhaps even impossible to repeatedly manually run a very large suite of regression tests that could be in the hundreds or thousands of tests so there is a big advantage to automating regression tests so that they can be run repeatedly and efficiently on a frequent basis. In an ideal case, automated regression tests can be run nightly to check the entire system as new functionality is being added. It would be very difficult, if not impossible, to do that frequently without automation for a complex system with a large number of individual regression tests to be run. And, of course, automating the regression testing frees up QA test resources to focus on more value-added tasks and also ensures that it is done consistently each time it is run.

Value-driven and risk-based testing

Another consideration in planning the testing strategy in any project is to recognize that it is impossible to test 100 percent of everything you could possibly test, and some method should be developed to determine what testing should be done and what testing can be minimized or eliminated. Fortunately, an agile project makes that easier to do because there is a lot more direct communications with the users to determine what is important and what is not important, and the user is directly engaged in testing the functionality of the software as it is being developed.

SUMMARY OF KEY POINTS

It's important for an agile project manager to have a broad, cross-functional view of all aspects of a project including development and testing practices to allow him/her to ensure that the people, process, and tools in the project are very well-integrated to maximize the overall efficiency and flow of the project and that they are also well-aligned with achieving the business goals the project is intended to accomplish.

Agile Software Development Practices

1. Continuous Integration and Code Refactoring

   An agile project calls for a different approach for managing the development process that should be well-integrated and well-aligned with the overall agile project management approach. It is not often possible to completely lay out and stabilize the design of a project early in

the project. A more dynamic approach for evolving the design as the project progresses is needed. Because the design is potentially changing throughout the project, a rigorous approach is needed to manage how changes are merged into the design, emphasizing continuous integration and code refactoring to ensure that the overall reliability of the design is still maintained.

2. Pair Programming

   Pair programming is sometimes used in agile projects to coach and mentor less-experienced developers and to provide higher levels of code quality and reliability. However, it is difficult to justify the economics of pair programming on all agile projects, and sometimes a more limited form of code reviews can accomplish some of the same goals.

3. Test-Driven Development

   Test-driven development is often used on agile project to simplify the development effort and to improve the reliability and quality of the software.

4. Extreme Programming

   Sometimes XP might be used to define the development process used with Scrum, but more often, people take more of an à la carte approach to selecting the development processes that they use with Scrum.

Agile Quality Management Practices

1. Key Differences in Agile Quality Management Practices

   The philosophy and approach for managing quality in an agile project is very different from a traditional project. The most important differences are that testing is an integral part of the development effort and is done concurrently with the development effort. The team, as a whole, needs to own responsibility for quality—it is not someone else's responsibility. The orientation is also more proactive towards preventing defects rather than a typical reactive approach of finding defects after the fact.

2. Definition of "Done"

   The definition of "done" is very important in an agile project. It provides a very well-defined standard of completeness that the team commits to. If it is properly done, it removes a lot of subjective judgment of whether a development effort is really complete or not.

3. The Role of QA Testing

   QA Testing plays a different role in an agile project. QA should be an integral part of the team and not a separate organization external to the team; however, that doesn't necessarily mean that developers should do their own testing. QA testing is an important skill and it is many-times worthwhile to have people on the team who are specialized and skilled in QA testing who are dedicated to planning and executing QA tests.

Agile Testing Practices

1. Concurrent Testing

   Instead of waiting until the end of the sprint to turn over all stories to QA for testing, a better practice would be to have the testers begin testing the software as soon as it is sufficiently complete for testing. That approach allows more concurrency of development and testing and avoids any last-minute surprises at the end of the sprint.

2. Acceptance Test-Driven Development

   Acceptance test-driven development is very similar to test-driven development but it is at a higher level of functionality. A big advantage of acceptance test-driven development is that it can eliminate the requirement for writing detailed functional specifications, it simplifies the requirements definition process, and it keeps everyone on the team clearly focused on the functionality that must be provided to satisfy the business need.

3. Repeatable Tests and Automated Regression Testing

   Because testing is done concurrently with development in an agile project and new functionality is being continuously added, it is essential for tests to be repeatable in an agile project to ensure that new functionality or other development activity has not broken something that was already tested and completed. As agile projects get larger, there is a big advantage to automating regression tests so that they can be run repeatedly and efficiently on a frequent basis. In an ideal case, automated regression tests can be run nightly to check the entire system as new functionality is being added.

4. Value-driven and Risk-based Testing

   It is impossible to test everything. Fortunately, an agile project makes that easier to determine what is important, and the user is directly engaged in testing the functionality of the software as it is being developed.

DISCUSSION TOPICS

Agile Software Development Practices

1. Why is it important for a project manager to have some understanding of development practices in an agile project?
2. What is continuous integration, and why is it important in an agile project?
3. What is test driven-development, and how is it different from acceptance test-driven development?

Agile Quality Management Practices

4. What is the biggest advantage of an agile quality management approach over a conventional non–agile testing approach? How is it different? What do you think it would take to make it work effectively?
5. What does the definition of "done" mean, and what is its importance in an agile project? Provide an example of a definition of done.
6. What is the role of a QA tester in an agile project? How is it different from a conventional, non–agile project?

Agile Testing Practices

7. Why is it important that tests be repeatable in an agile project?
8. Explain what automated regression testing is and why it is important in an agile project.

PART 2

Agile Project Management

AGILE IS CAUSING US to broaden our vision of what a *project manager* is and that will have a dramatic impact on the potential roles that a project manager can play in an agile project. In fact, the role of a project manager at a team level in a typical agile/Scrum project is undefined. That will cause us to rethink many of the things we have taken for granted about project management for a long time to develop a broader vision of what an *agile project manager* is. Some of the project management functions might be integrated into other roles and performed by someone who is not a dedicated project manager and doesn't have the title of project manager.

Chapter 6 – Time-Boxing, Kanban, and Theory of Constraints

The typical way of modeling traditional projects is based on a fairly statically defined model composed of work-breakdown structures, Pert charts, Gantt charts, etc. An agile approach is very different and is primarily oriented around managing the "flow" of requirements and tasks through an integrated development process. The tools that are most related to an understanding of managing "flow" are discussed in this chapter and include time-boxing, Kanban, continuous flow, and theory of constraints.

Chapter 7 – Agile Estimation

An agile estimation approach is very different from a traditional, plan-driven project estimation approach:

A traditional, plan-driven project estimation approach is based on attempting to accurately pin down the requirements, costs, and schedule for the project upfront before the project starts and it is like a contractual commitment with the customer.

An agile estimation approach is based on much higher levels of uncertainty and that level of uncertainty needs to be openly and transparently shared with the customer based on a collaborative spirit of trust and partnership with the customer rather than an "arm's-length" contractual relationship.

Chapter 8 – Agile Project Management Role

Agile is causing us to broaden our vision of what project management is, and that will have a dramatic impact on the potential roles that a project manager can play in an agile project. The image of a project manager is typically very heavily focused around a plan-driven development approach. That role will likely change dramatically in an agile environment and it may even eliminate the formal role of a project manager, as we've known it, in some situations. What is important; however, is not to preserve the role of a project manager, as we've known it, but learning to apply the discipline of project management in a much broader context and learning to blend traditional plan-driven principles and practices with more agile and adaptive principles and practices in the right proportions to fit a given situation.

Chapter 9 – Agile Communications and Tools

In my opinion, an agile project manager who doesn't know how to use one of the widely used agile project management tools like VersionOne [1] , Rally, Jira, or others is equivalent to a traditional project manager who doesn't know how to use Microsoft Project. Agile tools are fundamentally different from traditional project management tools like Microsoft Project. The fundamental difference is that traditional tools like Microsoft Project are heavily oriented around defining and managing the structure of project activities while agile tools are more oriented around managing the flow of project activities through a much more fluid structure.

Chapter 10 – VersionOne Tool Overview

I've chosen to use the VersionOne tool as a *representative example* of an enterprise-level agile project management tool. I chose VersionOne because it is a relatively complete set of the most important capabilities that I believe an agile project manager would need, it can be used with a variety of widely used development environments, and it is relatively easy to learn and use quickly.

Chapter 11 – Understanding Agile at a Deeper Level

An agile project manager needs to understand agile at a deeper level in order to apply it to different situations effectively. The key to that is to develop a *systems thinking* approach to understand agile principles behind the agile practices at a deeper level. In order to develop that kind of systems thinking approach, it is valuable to understand the roots of agile and how agile thinking evolved.

[1] VersionOne is a registered trademark of VersionOne, Inc.

6 Time-Boxing, Kanban, and Theory of Constraints

THE TYPICAL WAY OF MODELING traditional projects is based on a fairly statically defined model composed of work-breakdown structures, Pert charts, Gantt charts, etc. Figure 6.1 shows an example Gantt chart and Figure 6.2 shows an example Pert chart.

Those modeling tools work fine for a heavily plan-driven approach where the requirements as well as the overall design approach for meeting those requirements can be defined prior to the start of the project. However, they can be very difficult or impractical to apply in a more dynamic and adaptive project approach where the requirements are much more uncertain and difficult to define upfront. In that kind of project, it is very difficult and perhaps impractical to define and manage the structure of the project, and it becomes far more important to manage flow.

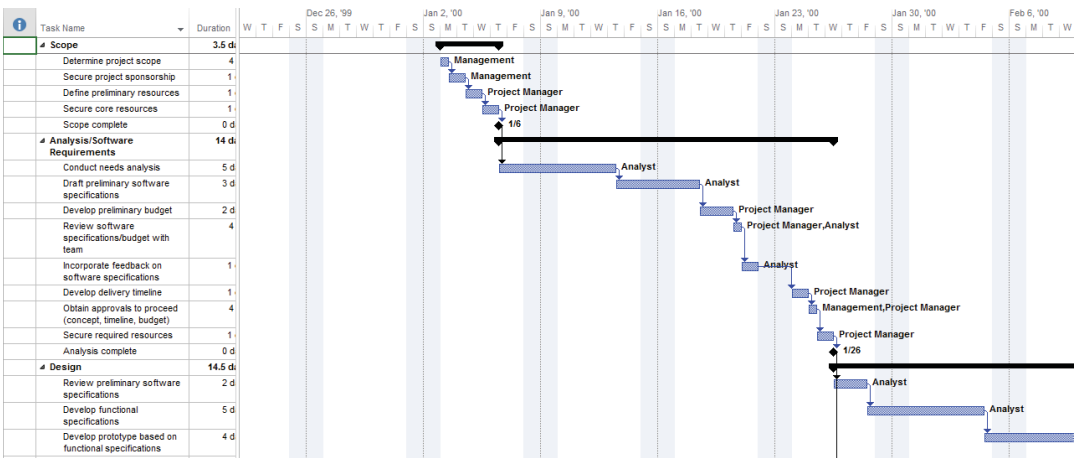


FIGURE 6.1 Example Gantt chart

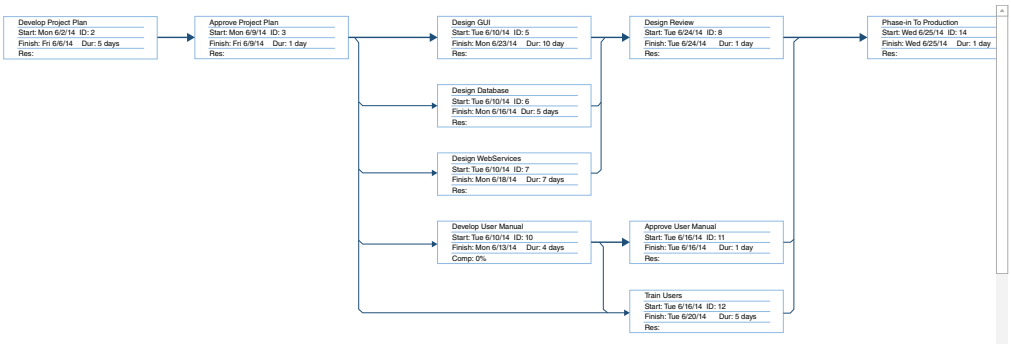


FIGURE 6.2 Example Pert chart

THE IMPORTANCE OF FLOW

Agile project management approaches are heavily based on a more dynamic model and on optimizing flow in order to maximize the efficiency and the throughput of the project team rather than statically managing the structure of the project tasks. The idea of *flow* originated with lean manufacturing and has been used in manufacturing to streamline production processes. Flow is one of the most important lean principles to understand to maximize the efficiency of any process. There are a number of factors that contribute to maximizing flow:

- *Small batch sizes*. Think of trying to push a number of balls through a pipe having a fixed diameter that limits the capacity you can push through the pipe. Large balls can block the pipe and cause bottlenecks while a greater number of smaller balls are more likely to flow smoothly through the pipe. For that reason, agile is heavily oriented on the idea of breaking up work into small chunks of effort to optimize flow.
- *Just-in-time production*. Maximizing the efficiency of a process is also dependent on having the right materials available in just the right amounts just in time to start production. In a manufacturing operation, it is wasteful to have huge inventories of raw materials waiting to start production, and it is also wasteful to not have a sufficient level of raw materials to start production. Either of those situations results in waste or inefficiency and doesn't contribute to maximizing flow. In a manufacturing process, flow is dependent on having the right raw materials that are needed available just-in-time to start production. In a product development process, requirements are equivalent to raw materials and flow is dependent on having requirements defined just-in-time to start development.
- *Concurrent processing*. Concurrent processing is another major factor that contributes to maximizing flow. If activities can only be done sequentially and there is a bottleneck in one phase of the process, it will inhibit the overall flow in the process as a whole. For example, if development and testing are done sequentially and there is a limited number of QA testers, the overall flow may be limited by the availability of testers. If, on the other hand, the process is designed around doing testing and development concurrently and those resources are more coordinated in a much more collaborative fashion, there is likely to be a much more efficient overall flow to the process.

This chapter will discuss several topics related to flow:

- *Time-boxing* plays a role in developing a cadence for optimizing the product development flow based on short-fixed length sprints. Having a regular cadence and a relatively stable velocity is essential for agile estimation.
- *Kanban* also provides an approach for optimizing WIP (work in process) flow.
- The *theory of constraints* is a well-known method developed by Eliyu Goldratt for resolving bottlenecks that may inhibit flow.

TIME-BOXING

Traditional project managers are used with an approach called schedule-boxing or scope-boxing, which means expanding the size of the schedule interval to accommodate the size of whatever task is being performed. That method might work if you can accurately size the scope of the requirements up front and adjust the schedule to fit the scope of the items but it has some significant disadvantages and is difficult to apply to a continuous flow-based model where the requirements are less-defined.

The alternative agile approach is breaking up tasks into small chunks that can be developed incrementally and using a fixed-length interval called a time-box for a sprint or iteration for development of those features. The scheduling approach with time-boxing becomes focused on "how many of these small, incremental features can be completed in a short fixed-length sprint?" rather than "how long does the development interval need to be to accommodate these features?" Naturally, to make this approach work, the individual features need to be small enough to fit inside of a fixed-length sprint.

Time-boxing advantages

There are a number of advantages of a time-boxing approach, which are summarized as follows:[1]

- *Focus:* The great advantage of time boxing is you learn how to focus your attention on the job at hand for the specified period of time.
- *Increased productivity:* When you set a timer and work diligently and in a focused manner on only the task you have identified, you work smarter and harder, and you get more done.
- *Realization of time spent:* When you use time blocking to get a job done, you realize how much time you might normally waste when working.
- *Time available:* Time-boxing makes you consciously aware of something you previously weren't consciously aware of—how much time you can give to a particular project.

Additional time-boxing productivity advantages

Beyond these advantages, there are a number of additional advantages of time-boxing that may not be immediately apparent. Time-boxing addresses two common productivity issues:

- *Parkinson's law* says, "Work expands so as to fill the time available for its completion." Fixing the time allowed eliminates wasted slack time that might be built into a scope-boxing approach.[2]
- *The student syndrome* refers to the phenomenon that many people will start to fully apply themselves to a task just at the last possible moment after a deadline. This leads to wasting any buffers built into individual task duration estimates.[3]

[1] "Timeboxing," Agile Hardware, http://www.agilehardware.com/pages/Timeboxing.html.
[2] "Parkinson's Law," http://en.wikipedia.org/wiki/Parkinson's_Law.
[3] "Student Syndrome," http://en.wikipedia.org/wiki/Student_syndrome.

Time-boxing can be applied to traditional projects as well as agile projects—it's a good development practice to keep a sustained pace of effort, but it is highly dependent on the culture and style of the organization. The general concept of time-boxing has many applications beyond its role in optimizing the flow of work in an agile project. In a broad sense, you can apply time-boxing to many different things to improve efficiency. For example, setting a time-box limit on the length of a meeting is often very useful to improve productivity and efficiency. Instead of letting a meeting go on-and-on without any limit (which can easily happen), it's useful to set a time limit for the meeting to keep the meeting focused on producing an acceptable end-result in a fixed amount of time.

KANBAN PROCESS

A *Kanban* process is a different kind of agile process. To understand *Kanban*, it is first necessary to understand the difference between a push process and a pull process.

Push and pull processes

A *push system* is totally planned in advance. An example would be a traditional manufacturing process:

- It starts with a manufacturing plan based on a *forecast* of demand that someone *thinks* will take place.
- Raw materials are ordered and held in inventory waiting to go into production.
- Production resources are planned in advance and allocated to meet the forecasted demand in the manufacturing plan.

In essence, raw materials are planned and queued and pushed through the manufacturing process. A traditional waterfall development process works on the same push principle. Requirements are forecast based on what someone thinks the customer needs, work is planned in advance, and resources are scheduled against the plan. Figure 6.3 shows a figurative illustration of what a push process looks like.

A *pull system*, on the other hand, is heavily demand-driven and less-planned. An example would be a manufacturing process where furniture is custom built to satisfy customer orders. In an ideal pull system:

- There may be a loosely defined plan to put capacity in place in anticipation of demand but no actual work is planned or scheduled until a customer order is received.
- Raw materials might be ordered only as needed to satisfy customer orders.
- Production resources would be allocated to work only as needed to satisfy customer orders.

This is the inverse of a push process—in a pull process, customer orders are queued and raw materials are *pulled* through the process to meet actual demand. Figure 6.4 shows a figurative illustration of what a pull process looks like.

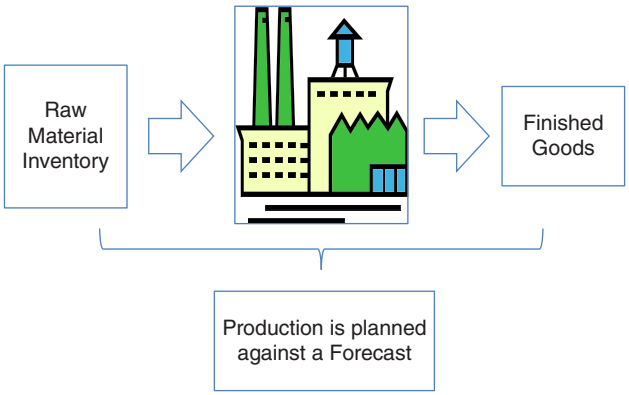


FIGURE 6.3 Push process illustration

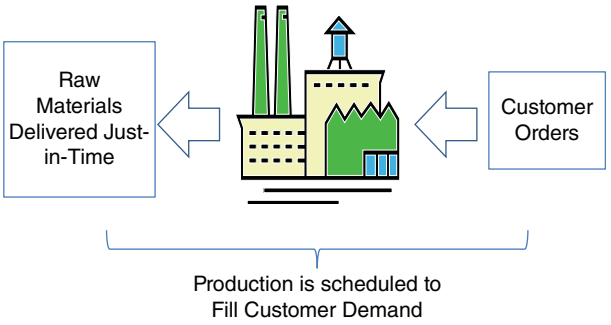


FIGURE 6.4 Pull process illustration

What is a Kanban process?

All agile processes are adaptive to customer demand to some extent rather than being totally planned; however, an ideal Kanban process is based totally on a pull process and is totally reactive to customer demand. The word *Kanban* is derived from the Japanese words *Kan,* meaning visual, and *Ban,* meaning card or board. The idea originated from inventory demand cards that are sometimes used in a manufacturing system. In an automotive assembly line, for example:[4]

- The person putting doors on cars only assembles enough doors to meet the demand for cars.
- When his inventory of doors runs low, he sends a demand signal (Kanban card) back to the previous operation that builds the doors.
- The person building the doors builds only enough doors to meet demand and he, in turn, sends a demand signal for more materials to build the doors his supply or materials runs low.

[4] "Kanban Development Oversimplified," AgileProductDesign.com (April 20, 2009), http://www.agileproductdesign.com/blog/2009/kanban_over_simplified.html.

In the real world, very few processes are totally unplanned (pull) and totally reactive. For example, in the manufacturing plant example, some amount of capacity needed to be planned to meet anticipated demand, but that capacity may be idle until it is needed to meet actual demand.

Outside of a manufacturing environment, Kanban processes are used primarily for demand-driven processes that are very reactive and difficult and/or impractical to plan. Examples that would be good candidates for a Kanban process include:

- A call service center queue for handling customer service calls
- Some repetitive development processes that are heavily demand-driven, such as a support function that creates reports for a business group
- An ongoing support process for providing continuous ongoing development of a product that includes bug fixes and minor enhancements

Kanban is sometimes used for software application development processes that need to be highly adaptive and may be difficult to plan in advance.

Differences between Scrum and Kanban

Table 6.1 shows a summary of the major differences between a Scrum process and a Kanban process.

A Scrum process is actually a hybrid of a push/pull process:

- Work is planned to some extent in the product backlog and broken up into sprints and releases (push).
- Within a sprint, the work is planned, limited, and allocated to the team (push).

TABLE 6.1 Differences between Scrum and Kanban

Typical Kanban Process	Typical Scrum Process
Kanban is generally based on a continuous flow model and work is not organized into sprints.	Scrum uses time-boxed sprints to organize work incrementally.
Items can be added to the work in process (WIP) anytime capacity is available.	Items cannot be added to work in process (WIP) once a sprint is in progress.
Estimation of work is optional in a Kanban process.	The team estimates work at the beginning of each sprint and plans sprint capacity based on expected velocity.
WIP can be organized into stages with WIP limits to manage flow.	A similar approach may be used if desired; however, there is less of a need to manage WIP limits in Scrum.
A Kanban board can be shared by multiple teams.	The sprint backlog is owned by a single team.
No prescribed roles—a Kanban process can be adapted to almost any kind of team structure.	Scrum uses prescribed roles (Scrum Master, product owner, team).

- However, it is demand-driven based on priorities set by the product owner, and the product owner has considerable ability to influence what is built (pull).

In one sense, it is a pull process because a certain amount of capacity is made available in the team to develop requirements and there is not a large backlog of requirements, as there would be in a waterfall process.

- Requirements in a Scrum process are typically groomed to get ready for development more on a just-in-time basis, which is similar to a pull process.
- On the other hand, a Scrum process can be somewhat of a push process because there is nothing to prevent planning high-level requirements in advance of development in order to estimate the project cost and schedule at a high level.

A pure Kanban process is totally demand-driven (pull) and not planned:

- Only a limited effort is made to plan requirements in advance—for example, in planning a customer service response capability, it's impossible to predict the exact number and nature of the customer service calls that might come in but a limited amount of planning is needed to in order to do capacity planning and pipeline planning to put the appropriate capability in place to handle the anticipated volume of calls.
- A Kanban process does not break up the work into sprints at all.
- It is a continuous flow model—work is taken into the process immediately based on priority as soon as resources are available to work on it.

The best example of a Kanban process is something like a customer service call queue. A certain number of customer service representatives are hired to answer calls in the expectation of customer demand, but the actual work in the process is totally reactive to customer demand and the utilization of the process is dependent on the level of demand that comes in and the time it takes to handle each call.

Work-in-process limits in Kanban

A critical concept in Kanban systems is the idea of WIP. Kanban processes many times have stages that work goes through—an example is a customer service system designed to solve customer problems. That kind of system typically has different levels of support, depending on the nature and severity of the customer's problem, as shown in Figure 6.5.

A Kanban system is designed to optimize the overall flow through the system and each stage in the process may have capacity limitations that limit the overall work in process. For example, in a typical customer service response system, there is many times an initial level that is designed to screen calls and weed out problems that can be easily resolved without requiring more sophisticated resources. Only problems that cannot be resolved in that stage are routed on to a second or third level of support for further resolution depending on the complexity and difficulty of the problem.

In this situation, the size of the screening staff would typically be much larger than the more sophisticated problem resolution staff to optimize the flow and efficiency of the overall process.

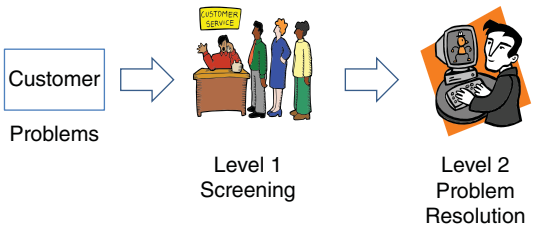


FIGURE 6.5 Levels of support in a customer service response system

Each stage has a WIP limitation based on the capacity of the resource capacity in that stage and the flow through the process will be determined by which stage in the process is more limiting on the overall flow.

Kanban boards

Kanban boards are tools that are used to visually show the flow of items through a Kanban process. For example, Scrum and other agile processes use Kanban boards to manage the flow of items in a sprint. The Kanban board can be as simple as a white board with stickies on it or 3 × 5 cards that are manually moved around the board to show progress through the flow.

Kanban boards can also be implemented using an automated online tool to track progress as shown in Figure 6.6.

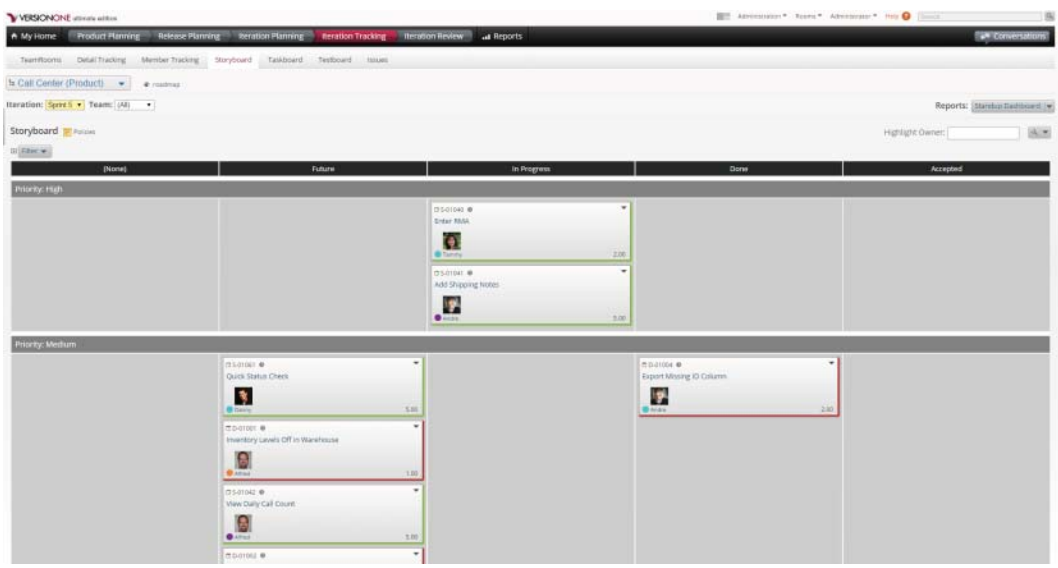


FIGURE 6.6 An example Kanban board in an online tool

THEORY OF CONSTRAINTS

All agile processes (especially Kanban) emphasize optimizing the flow through the entire process—bottlenecks can develop in each stage of the process, and a key concept that is extremely useful for optimizing the overall flow is the theory of constraints that was originally published by Eliyu Goldratt in his book *The Goal* in 1984.[5] It involves five steps that are repeated over and over until you finally optimize the flow through the process:

1. *Identify*. The first step is to *identify* the stage or portion of the process that is the most critical constraint or bottleneck. For example, in the customer service system, it may be that there are not enough people to handle calls and do the initial screening and, as a result, people are waiting for a long time to get through.
2. *Exploit*. The second step is to do whatever you can to *exploit* or optimize that portion of the process to make it more efficient. For example, in the customer service response system, you might add an improved interactive voice response system that would provide some initial screening to relieve the load on the people answering the calls.
3. *Subordinate*. The third step is that, after you've done whatever you can to optimize the performance of that constraint, you should *subordinate* everything else in the process to work within that limitation. For example, if the number of people available to answer the phone is the limitation, there is no point in having an excessive number of people in the second stage to resolve problems if those people are idle waiting for calls to come through.
4. *Elevate*. If the process flow through the overall system is still not sufficient, the next step is to do whatever is necessary to *elevate* the constraint. For example, in the customer service response system, you might add additional resources to answer the phone until that portion of the process is no longer the limiting constraint.
5. *Check for new constraint*. Finally, once you've relieved a bottleneck, another part of the process then will typically become the bottleneck. For example in the customer service response system, if you have relieved the constraint of the front end people answering calls, the people on the back-end resolving problems may then become the bottleneck and you need to repeat the above steps with that as the new primary constraint.

Here's an example. In a company I worked in, we had over 100 legacy applications that needed to be upgraded to a new version of software, and we designed a Kanban flow with a number of different stages that each application would go through:

1. A senior-level technical lead would check out the application and do a preliminary survey of the application to identify any risks in the conversion process and flag those items to the project team. He would also establish coordination with the owners of that application and any other related applications to begin the conversion process.

[5]Eliyu Goldratt, *The Goal: A Process of Ongoing Improvement* (Great Barrington, MA: North River Press, June 2012).

2. An offshore project team was assigned to do the conversion process and to perform initial unit testing.
3. The application was turned over to quality assurance (QA) for formal QA testing.
4. The application was staged for release to production and released.

The process flow for that process is shown in Figure 6.7.

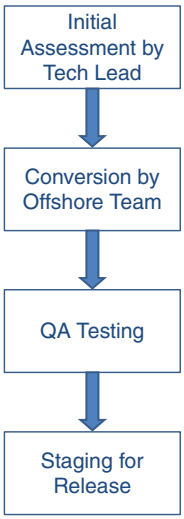


FIGURE 6.7 Example process for theory of constraints analysis

We were given a goal to convert more than 100 applications in about a year, and the plan was to do about 6 to 7 applications per month. We explored a number of options to try to accelerate this process; this is a good example of how the theory of constraints can be used:

1. *Identify*. The first step in the process is to *identify* the system constraint or bottleneck that is most critical for limiting the flow. In this situation, the system constraint was the resources available to perform the QA testing. There were a limited number of QA resources to do the testing that was required, and they would become quickly saturated with testing because the conversion effort wasn't that difficult for many of the applications and some of it could be automated but the testing effort was significant to ensure that the application functionality wasn't impacted by the conversion.
2. *Exploit*. The next step is to *exploit* the constraint. Given that the QA resources were the bottleneck, what can be done to optimize the use of those resources to improve the overall flow? We were able to do several things to optimize those resources, including planning the conversion process to spread the load among the QA resources we had as evenly as possible.
3. *Subordinate everything else*. Once we had done everything we could to do optimize the QA resources, it made no sense for the conversion process to go faster than QA could handle the

applications. As a result, the application conversion effort needed to be slowed down to match the speed of the QA process. Otherwise, a very large queue of applications waiting to be tested would have piled up waiting for QA.

4. *Elevate*. The next step to improve the flow would be to add more QA resources to *elevate* the constraint and reduce the bottleneck. In this particular situation, that was not an option.
5. *Check for new constraint*. Of course, if you add enough QA resources, at some point that will no longer be the most critical constraint and the bottleneck will move somewhere else in the process, such as the number of developers to do the conversion process. (We never really got to that point in this particular example.)

SUMMARY OF KEY POINTS

1. The Importance of Flow

   The idea of *flow* originated with lean manufacturing and has been used in manufacturing to streamline production processes. Flow is one of the most important lean principles to understand to maximize the efficiency of any process.

2. Time-boxing

   Time-boxing, as opposed to schedule-boxing or scope-boxing, is a more efficient way of managing flow in an agile project for a number of reasons, and can result in higher productivity. It's equivalent to passing work to be done through a pipeline and using uniform size containers to handle the elements in the flow. It provides a way of establishing a regular cadence on an agile project, as well as measuring velocity, which can be used to measure the efficiency of the process and predict future performance.

3. Kanban Process

   Kanban is primarily a continuous flow process. It is primarily a pull process that is totally demand-driven. Rather than work being planned and broken up into sprints, the work is taken in continuously from a prioritized queue of work to be done. The best candidates for Kanban are processes that are reactive and difficult to plan in advance. For example, managing the workload in a queue of customer service requests would be a good candidate for a Kanban process. Although a Kanban process and a Scrum process are different, many of the principles related to managing flow are the same in both processes.

   Kanban boards are widely used in Scrum and many other processes as a visual way of tracking progress through a process.

4. Theory of Constraints

   The theory of constraints is an approach for analyzing problems related to continuous flow. It provides a systematic way of identifying and removing bottlenecks to improve the overall flow of a process.

DISCUSSION TOPICS

The Importance of Flow

1. What are the factors that contribute to maximizing flow?

Time-boxing

2. What is time-boxing? What is the alternative? Why is time-boxing generally considered a better approach for agile projects?
3. When would it not work so well?

Kanban Process

4. What are the most important differences between a Kanban process and Scrum? When would a Kanban process make more sense than a Scrum process?
5. What is a Kanban board? How is it used in Scrum?
6. Explain how WIP limits affect a Kanban process.

Theory of Constraints

7. Discuss an example of a problematic project situation and how you might have applied the theory of constraints to analyze the problem and find an appropriate solution.

7 Agile Estimation

AGILE ESTIMATION OVERVIEW

What's different about agile estimation?

BEFORE DISCUSSING AGILE ESTIMATION, it's important to have an understanding of what's different about the environment for an agile project, because there are some very significant differences that affect how you would do any kind of estimation.

A traditional plan-driven or waterfall project is usually based on somewhat of a contractual relationship between the business users and the project team. Typically, the business users agree to some fairly well-defined requirements and sign off on them prior to the start of the project. The project team then commits to a cost and delivery schedule to meet those requirements and any changes in the scope of the requirements are controlled from that point forward. In order to manage the reliability of the estimates, some kind of disciplined process for managing changes to the requirements is implemented, and significant changes are considered to be an exception rather than the norm. If a significant change does takes place, it normally triggers an assessment to determine the impact on the costs and schedule of the project and any initial estimates are recalculated and revised if necessary.

An agile project has a very different model:

- First, the requirements are typically not necessarily well-defined in detail up front; and, in addition, change is the norm rather than the exception. Change is expected and encouraged as the project is in progress. Naturally, any estimates can only be as good as the requirements are defined and any change in the requirements might invalidate any initial estimates. For that reason, some people say that it is futile to even attempt to estimate the costs and schedule of an agile project because the requirements are only going to change. I don't necessarily agree with that point of view—just because a project has some level of uncertainty, doesn't mean it is totally futile to estimate the costs and the schedule; but, of course the level of accuracy in the estimate needs to be

understood in the context of the level of uncertainty associated with the project. Very few people get a blank check to do an agile project without any expectations of what the cost and schedule might be.

- An agile project is based on a much more collaborative partnership between the business users and the project team and there is a much higher level of transparency and sharing of information related to the project, which should lead to a much higher level of mutual understanding of the risks and uncertainties in the project as well as a shared understanding of the level of accuracy in any cost and schedule estimates.

The most important thing is to properly set the expectations of the customer regarding whatever estimates are made:

- A traditional plan-driven project estimate might have been based on what was thought of as a relatively static and well-defined model of what the requirements are and how the project will be managed (which might or might not have been realistic).
- An agile estimate is typically based on a high-level and dynamic view of what the requirements are that is understood to be not extremely accurate and likely to change.

Naturally, those are not discrete, binary alternatives and there are many variations in that involve a blend of those two approaches.

Table 7.1 shows a summary of the differences in these two environments.

TABLE 7.1 Estimation in Agile vs. Traditional Processes

	Typical Plan-driven or Waterfall Estimation Process	**Typical Agile Estimation Process**
Relationship of customer or business user to the project team	■ Arm's-length contract based on firmly-defined and signed-off requirements ■ Limited transparency and visibility of the user to project details and issues	■ Collaborative joint partnership model ■ High levels of transparency and sharing of information ■ Mutual understanding of the relationship of uncertainty and flexibility to estimates
Control over changes	Changes to requirements are controlled, and any significant changes are treated as an exceptional situation	Changes to requirements are encouraged and are the norm rather than an exception

Developing an estimation strategy

There are two common problems associated with estimation:

1. Analysis paralysis:
   - Spending too much time attempting to develop detailed estimates based on highly uncertain and unreliable information
   - Delaying making commitments or delaying making any decision at all until the information required to make the decision has a very high level of certainty about it and all of the risks associated with a project are well-known and completely understood
2. Cavalier approach:
   - Not worrying about managing uncertainty and risk at all and just starting to do a project with very little or no upfront planning and estimation
   - Assuming that whatever uncertainty and risk is inherent in the project will be discovered and somehow work itself out as the project proceeds

The right approach for a given project is somewhere in between those extremes and the approach selected needs to be consistent with the level of uncertainty in the project:

- The level of uncertainty in a project can vary widely, from a pure research and development initiative with very high levels of uncertainty to projects with very low levels of uncertainty (like building a bridge across a river).
- It is foolish to underestimate the level of uncertainty in a project and create unrealistic estimates based on very uncertain information.
- However, it can be equally foolish to not take advantage of information you do know and use it to make informed decisions.

The key is to honestly and objectively assess the level of uncertainty and risk in the project and reach agreement among all participants in the project (customer and project team) on an estimation approach that is consistent with that assessment.

Management of uncertainty

Being able to deal with uncertainty effectively is a very critical skill for an agile project manager. You have to be able to accurately assess the risks and uncertainty in a given situation and make informed decisions based on the best information that you have, knowing that it might not be a perfect decision. That is not really a new skill with agile—it is a skill that all project managers should have; however, it is especially important in an agile environment where the level of uncertainty may be significantly higher.

This process of managing uncertainty is similar to making a decision when you're betting on a card game. Many people are familiar with the game of blackjack where players

FIGURE 7.1 Example of a blackjack hand

compete against the dealer to get the highest value of cards without going over 21 points (see Figure 7.1):

- Face cards are worth 10 points each and aces can be worth either 1 or 11 points each.
- Players are each dealt two cards face up.
- The dealer is dealt two cards (one face up and one face down).
- The player has to decide to take more cards (called taking a hit), risk going over 21 points, and automatically losing; or to stay with his/her current cards and risk losing to the dealer because the dealer has a higher number of points.

Here's how I would go about analyzing this situation:

1. What information do I know for certain?
   - I have 16 points in my hand.
   - The dealer has more than 10 points in his/her hand.
2. What information is unknown or uncertain?
   - What is the dealer's second card?
   - What card will I be dealt next?
3. What is a reasonable assumption to make?
   - There is a high probability that the dealer's second card is another 10 or a face card, giving the dealer a total of 20.
   - If that is true, I will lose on 16 if I don't take a hit, so I should risk taking another card.

This is an example of making an informed decision based on incomplete information. It is something a gambler does all the time. It is a very important skill for an agile project manager. An agile project manager needs to be frequently making informed decisions in a very uncertain environment.

AGILE ESTIMATION PRACTICES

Levels of estimation

In an agile project, there is no prescribed way to do estimation, and the exact approach could range from doing very little or no estimation at all to having several levels of planning and estimation based on the scope and complexity of the project, as shown in Figure 7.2.

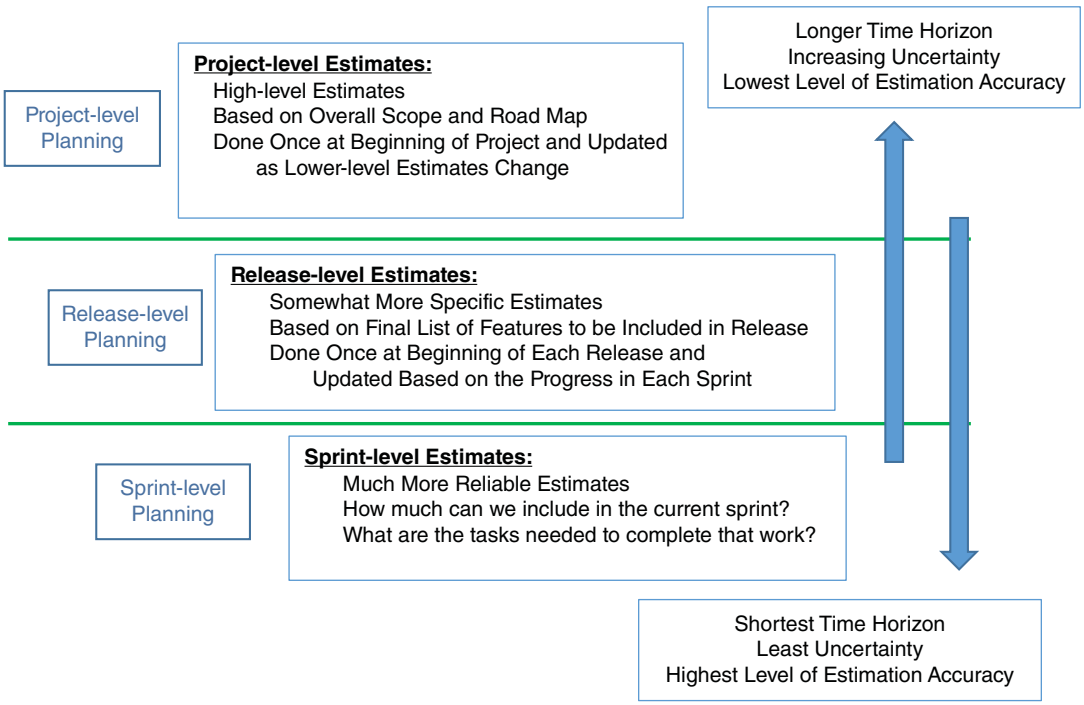


FIGURE 7.2 Levels of agile estimation

The level of emphasis that you would put on each of these levels will also vary significantly, based on the nature of the project. Doing sprint-level planning and estimation is most essential, but any of the levels of planning and estimation just described are optional, depending on the nature of the project. For example, an agile estimation approach could be as simple as having sprint-level estimates only without any project-level or release-level estimates at all.

The approach is totally dependent on the level of uncertainty in the project and the level of predictability and control that is desired. Of course, the level of predictability and control needs to be consistent with the level of uncertainty in the project. Attempting to achieve a very high level of predictability and control in a project that has very high levels of uncertainty may be futile.

Each of these levels of planning would naturally have a different planning horizon, different levels of uncertainty, and a different level of accuracy for any estimates associated with it. Naturally, the longer the planning horizon, the more the uncertainty will be and the less accurate the estimates will be.

Sprint-level planning is the lowest level, has the shortest planning horizon, and it should have the highest level of certainty and the highest accuracy in the estimates. An experienced agile team should be able to accurately predict the capacity that can be completed in a given sprint after the team has reached maturity and the velocity of the team has begun to stabilize.

Before the team starts a sprint, the features to be included in that sprint should be known, and they should be understood well enough to make a reasonably accurate estimate of the level of effort required to complete those features. The details associated with those features may be

further elaborated during the sprint, but features should not normally be added or subtracted from the items to be included in the sprint while the sprint is in progress.

Project-level planning is the highest level of planning, has the longest time horizon, and has the highest level of uncertainty. Naturally, the estimates at this level can only be as accurate as the level of certainty in the requirements, so the estimates may not be very accurate. The level of accuracy depends on a number of factors such as the nature and complexity of the project, the level of uncertainty in the project requirements, and the importance of having schedule and cost estimates.

Release-level planning is an intermediate level of planning between project-level planning and sprint-level planning and is optional, based on the scope and complexity of the project. It can be very useful for large, complex projects to break up the project into releases.

What is a story point?

A *story point* is a metric commonly used for estimation in agile projects. It is a way of sizing the level of effort associated with a particular user story. A story point is normally not directly tied to any specific measure of time; it is simply a relative measure of the level of difficulty of a particular story relative to other stories, and a story point may not have the same meaning among different agile teams.

For example, as I was developing the course materials for the agile project management course based on this book, I used story points to estimate the level of effort required for each lesson. One story point was equivalent to one simple PowerPoint slide, and I used that as a reference point for sizing the level of effort associated with other slides and lessons, which is a relative sizing estimate.

Jeff Sutherland has written a great article to explain why story points are better than hours for agile estimation:

> Story points give more accurate estimates, they drastically reduce planning time, they more accurately predict release dates, and they help teams improve performance. Hours give worse estimates, introduce large amounts of waste into the system, handicap the Product Owner's release planning, and confuse the team about what process improvements really worked . . .
>
> The stability of a user story is critical for planning. A three point story today is three points next year and is a measurable part of the product release for a Product Owner. The hours to do a story depend on who is doing it and what day that person is doing it. This changes every day.[1]

Story points are also a more relevant metric to measure completion of work: "Hours completed tell the Product Owner nothing about how many features he can ship and when he can ship them."[2]

[1] Jeff Sutherland, "Story Points: Why are they better than hours?," posted on May 16, 2013, http://scrum.jeffsutherland.com/2010/04/story-points-why-are-they-better-than.html
[2] Ibid.

In reality, it is very difficult to start a new agile team using story points because they have no frame of reference to compare it to. In that case, I have sometimes made an arbitrary decision to let one story point be approximately one man-day of work but that is just an arbitrary decision to get started with. As the team gains more experience, they should be able to make relative comparisons among stories based on their past experience without a reference to any specific measure of time. An alternative to story points for teams who are not very experienced is *ideal days*. An ideal day is defined as follows:

> A unit for estimating the size of product backlog items based on how long an item would take to complete if it were the only work being performed, there were no interruptions, and all resources necessary to complete the work were immediately available.[3]

Once a team gains experience, story points become very easy and intuitive.

> A mature agile team intuitively knows what a story point means in terms of the relative size of a user story compared to other stories that it has sized in the past, but how does a new team that perhaps even has people who are new to agile get started with story points? As I mentioned in another blog post, Ideal Days is a story point estimation scale that blends size with effort and degrades the backlog sizing process into a drawn-out time estimation exercise.[4]

A Fibonacci sequence is often used for story point estimates because it provides an appropriate level of discrimination between large and small estimates without attempting to very accurately discriminate large estimates. A Fibonacci sequence is a series of numbers of the form of (1, 2, 3, 5, 8, 13, 21—where each number is the sum of the previous two numbers). T-shirt sizes can also be used to create estimates. Both story points and T-shirt sizes are designed to provide a very imprecise framework for making estimates that is appropriate to an agile environment.

For an estimation approach based on story points to work with agile teams, it is important for the teams to be stable. If people are rotated in and out of an agile team, it will destabilize the team and make the estimation process very difficult.

How are story points used?

Story points are used in different ways based on the level of planning and estimation being done.

1. High-level product backlog grooming:
   - In grooming the product backlog, stories are estimated in story points.
   - A voting technique is used to estimate the level of effort associated with each story in story points.

[3] Ideal day definition, http://www.innolution.com/resources/glossary/ideal-day.
[4] Blog archives, "Getting Started with Story Points," posted by Alec Hardy, Feb. 3, http://agilereflections.com/tag/ideal-days/.

- The relative size of the story points provides some information to the product owner to assess the business value against the level of effort.
- Based on that assessment, the product owner will rank order the items in the product backlog to try to maximize the business value to be gained.

2. Tactical sprint planning:
   - The project team determines the team's velocity in story points based on the number of story points completed in each sprint.
   - At the beginning of each sprint, the project team will make a determination of how many stories that they can take into the current sprint based on their previous velocity history.
   - In sprint planning, a more detailed estimate of the level of effort required for the tasks associated with each story may be used to validate the story point estimates.
3. Project-level and release level planning:
   - High-level story point estimates, combined with team velocity estimates, may also be used for developing project-level and release-level estimates.
   - For example, to develop a rough estimate of the time required to complete a project, it would be necessary to make a high-level estimate of the story points required for each story to be included in the project or release, total those story points, and then divide that by the velocity of the team in story points per sprint to determine how many sprints it will take to complete the project or release.

What is planning poker?

Planning poker is a commonly used approach to come up with story point estimates:

- A story is presented to the team and is discussed to resolve any significant uncertainties or questions.
- Each member of the team chooses a card representing his or her estimate in story points but does not reveal the estimate to others.
- When everyone is ready to vote, all members of the team show their cards.
- The team then discusses why they voted differently to try to converge on a single estimate that everyone agrees to.

Planning poker is based on the Delphi method. The Delphi method is a structured communication technique, originally developed as a systematic, interactive forecasting method that relies on a panel of experts coming to consensus on an estimate:

- The experts answer questionnaires in two or more rounds.
- After each round, a facilitator provides an anonymous summary of the experts' forecasts from the previous round as well as the reasons provided for the facilitator's judgments.

- Thus, experts are encouraged to revise their earlier answers in light of the replies of other members of their panel.
- It is believed that during this process the range of the answers will decrease and the group will converge towards the 'correct' answer. Finally, the process is stopped after a predefined stop criterion (e.g., number of rounds, achievement of consensus, and stability of results) and the mean or median scores of the final rounds determine the results.[5]

An online tool is available for implementing planning poker at the following website: www.planningpoker.com.

VELOCITY AND BURN-DOWN/BURN-UP CHARTS

The estimation process in an agile project should be dynamic and continuously refined as the project progresses. An agile estimate is based primarily on the velocity of the team, projecting that velocity into the future, monitoring the actual velocity against the projected velocity, and then adjusting the estimate as necessary.

Velocity

The concept of velocity is essential to agile estimation. Velocity is defined as follows:

> Velocity is the measure of the throughput of an agile team per iteration. Since a user story, or story, represents something of value to the customer, velocity is actually the rate at which a team delivers value. More specifically, it is the number of estimated units (typically in story points) a team delivers in an iteration of a given length and in accordance with a given definition of "done."[6]

Here is an example of how velocity can be used to develop an estimate for completing future work:

- Assume that a team has completed an average of 60 story points of work over the past three sprints, and each sprint is two weeks long.
- Suppose there is a total of 300 story points' worth of work to be completed in the current release, and the product owner wants to estimate when the remainder of the release will be completed.
- An estimate for completing the remaining in the release would be 300/60, or five sprints, and since each sprint is two weeks long, it will take approximately 10 weeks to complete the work remaining in the release.

[5] "Delphi Method," http://en.wikipedia.org/wiki/Delphi_method.
[6] "Velocity," Agile Sherpa, http://www.agilesherpa.org/agile_coach/metrics/velocity/.

Burn-down charts

Of course, any estimate is only an estimate, and it should be continuously adjusted as the project progresses as more becomes known about the actual velocity of the team and the level of effort remaining to be completed. A way of doing that is to measure the ongoing actual velocity of the team against the projected velocity and adjust the projected velocity based on actual results. An important tool for doing that is a burn-down chart:

> A burn-down chart is a chart often used in scrum agile development to track work completed against time allowed. The *x*-axis is the time frame, and the *y*-axis is the amount of remaining work left that is labeled in story points and man hours, etc. The chart begins with the greatest amount of remaining work, which decreases during the project and slowly burns to nothing.[7]

Figure 7.3 shows an example of a burn-down chart. In this chart, the *y*-axis shows the total work remaining to be completed in story points, which is 120 units at the beginning of the sprint and is

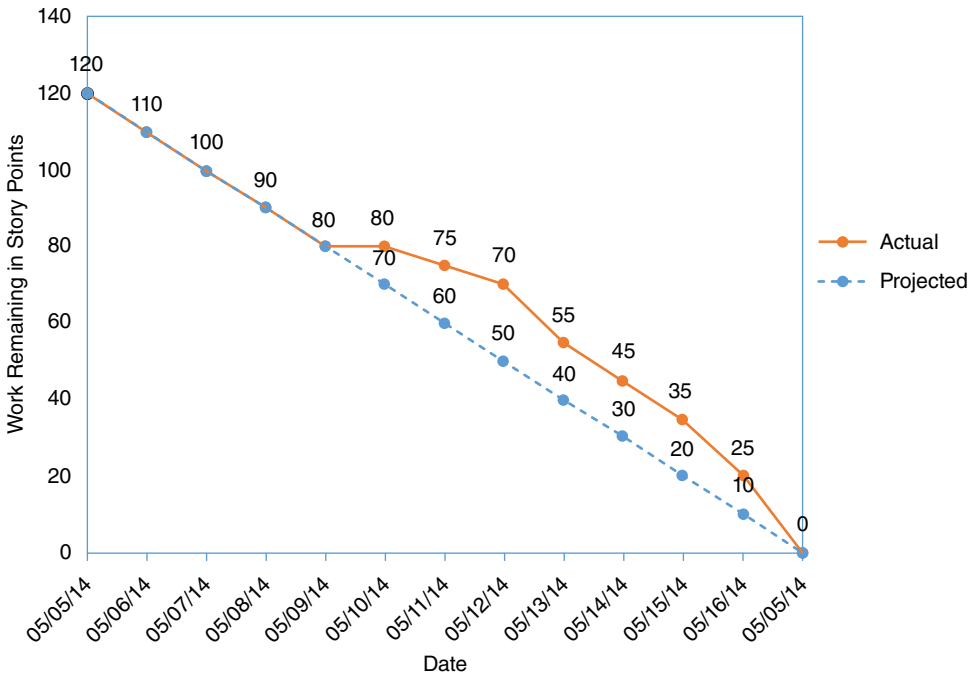


FIGURE 7.3 Example burn-down chart

[7] "What is a Burn-down Chart?" Techopedia, http://www.techopedia.com/definition/26294/burndown-chart.

projected to go down to zero by the end of the time interval. The time interval being measured could be a sprint, a release, or an entire project. The diagonal line between those two points shows the projected rate of completing work in the time interval. The darker solid line shows the actual performance of the team as the sprint is in progress:

- If the actual work remaining is above the projected line, work is proceeding slower than expected.
- If the actual work remaining is below the projected line, work is proceeding faster than expected.

In Figure 7.3 the work started out very close to the projected schedule, then started to lag behind and finally caught up with the projected schedule by the end of the sprint. Burn-down charts are very useful for monitoring the actual progress against projected progress to continuously adjust the estimated completion of an effort.

Burn-down charts are typically measured in either hours of work remaining or story points of work remaining. Each of those metrics provides different information:

- Hours of work remaining is difficult to track because it requires the people on the team to report time spent on tasks, as well as estimating the hours of work remaining, which can be difficult to do. Estimates of hours of work remaining may not be very accurate.
- Story points are easier to track, and it's an all-or-nothing metric—the team gets full credit for all of the story points associated with a given story if the story has been complete and meets the definition of "done," or no credit if it has not been completed. That's probably a more reliable and more accurate metric.

Burn-up charts

A burn-up chart is essentially the mirror image of a burn-down chart:

> A burn-up chart is a graphical representation that tracks progress over time by accumulating functionality as it is completed. The accumulated functionality can be compared to a goal such as a budget or release plan to provide the team and others with feedback. Graphically the X axis is time and the Y axis is accumulated functionality completed over that period of time. The burn-up chart like its cousin the burn-down chart provides a simple yet powerful tool to provide visibility into the sprint or program.[8]

Figure 7.4 shows an example of a burn-up chart. One of the advantages of a burn-up chart is that it provides a way to show a change in the scope of the projected work. In this example on May 13, 2014, there was an increase in the scope of work remaining from 80 to 100 story points, which raised the end goal from 120 to 140. (That shouldn't happen in the middle of a sprint, but it can

[8]Thomas F. Cagley, "Metrics Minute—Burn up Charts," May 9, 2011, http://tcagley.wordpress.com/2011/05/09/metrics-minute-burn-up-charts/.

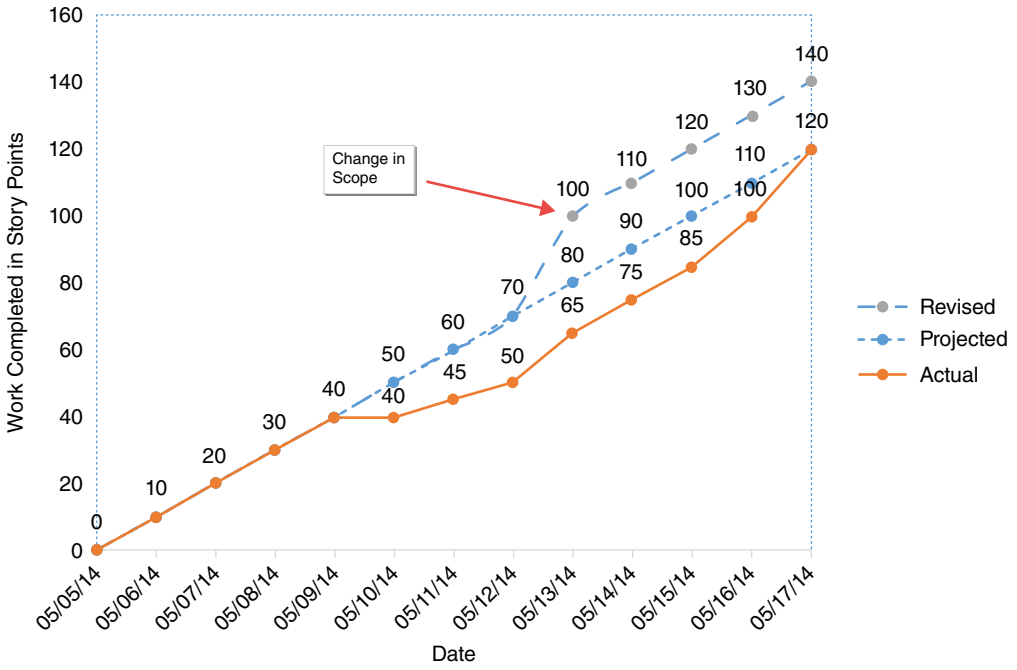


FIGURE 7.4 Example burn-up chart

easily happen in the middle of a release or project.) The chart shows that the team finished the sprint on the original goal of 120 but failed to meet the revised goal of 140.

SUMMARY OF KEY POINTS

1. Agile Estimation

Agile estimation is based on a very different environment than traditional project estimation approaches.

- A traditional model is based on somewhat of a contractual relationship between the customer and the project team and the project estimates are expected to be reasonably accurate to support that kind of relationship.
- Any agile project estimate can only be as accurate as the requirements are known and there typically is lot of uncertainty in the requirements that would limit the accuracy of any estimate. For that reason, it is essential for an agile project to be based more on a collaborative partnership between the business users and the project team as well as a shared understanding of the level of accuracy in any cost and schedule estimates.
- Management of uncertainty is an important skill for an agile project manager—it is something that all project managers should have, but it is especially important in an agile environment where the level of uncertainty may be significantly higher.

2. Agile Estimation Practices

There is no prescribed way to do estimation in an agile project and the exact approach could range from doing little or no estimation at all to having several levels of planning and estimation based on the scope and complexity of the project. Whatever level of estimation is selected, the approach should be consistent with the level of uncertainty in the project and mutually understood by all participants in the project.

Agile estimation practices are based on an understanding of velocity and flow and require breaking up requirements into small functional elements typically in the form of user stories that can be sized by the project team in terms of story points to estimate their difficulty.

A *story point* is a metric commonly used for estimation in agile projects. It is a way of sizing the level of effort associated with a particular user story.

3. Velocity and Burn-down/Burn-up Charts

An agile estimate is based primarily on the velocity of the team, projecting that velocity into the future, monitoring the actual velocity against the projected velocity, and then adjusting the estimate as necessary.

Burn-down charts are typically measured in either hours of work remaining or story points of work remaining. Each of those metrics provides different information. A burn-up chart is a graphical representation that tracks progress over time by accumulating functionality as it is completed.

DISCUSSION TOPICS

Agile Estimation

1. How would you go about determining the appropriate estimation strategy for a project?

2. Discuss an example of a real world project and how you might have done it differently with a more agile estimation approach.

3. What is the advantage of story points over estimating in hours?

4. Complete the agile estimation exercise on the following page and answer the following questions:

- Select two to three of the items in the list and write a user story to describe each one.
- If the work is divided into two-week sprints and I have a velocity of completing 20 story points of work in each two-week sprint, how long will it take to complete this work? (Note that the items in the hierarchy are summed at different levels.)

Agile Estimation Exercise

5. I need to develop an estimate for the schedule required to develop course materials to support this book. I have identified the following tasks and estimated the work associated with each task in story points.

List of Tasks for Estimation

Title	Estimate
1. Introduction, Course Objectives, and Agile Overview	8.00
Introduction and Course Objectives	2.00
Introductions	1.00
Course Objectives	1.00
Agile Overview	6.00
What Is Agile?	3.00
Agile Perception versus Reality	3.00
2. Agile Fundamentals	17.00
Agile History, Values, and Principles	11.00
Agile Manifesto Values	3.00
Agile Manifesto Principles	8.00
Agile Benefits and Obstacles to Becoming Agile	6.00
Agile Benefits	3.00
Obstacles to Becoming Agile	3.00
3. Scrum Overview	34.00
Scrum Roles	3.00
Scrum Master Role	1.00
Product Owner Role	1.00
Team Role	1.00
Kanban Process Overview	10.00
What Is Kanban?	1.00
Differences Between Push and Pull Processes	2.00
Differences Between Kanban and Scrum	2.00
WIP Limits in Kanban	1.00
Theory of Constraints	2.00
Kanban Boards	2.00
Scrum Methodology	8.00
Time-Boxing	3.00
General Scrum/Agile Principles	10.00

8 Agile Project Management Role

AGILE IS CAUSING US TO broaden our vision of what *project management* is, and that will have a dramatic impact on the potential roles that a project manager can play in an agile project. The image of a project manager is typically very heavily focused on a plan-driven development approach. If you read many books on project management, you will find a lot of discussion on how to use:

- Project plans
- Work breakdown structures
- Pert charts and Gantt charts
- Microsoft Project to plan a project

All of those things are good things for a traditional project manager to know, but many of them become irrelevant in an agile environment, which calls for a very different approach. In an agile environment, you may not have enough information upfront to develop detailed project plans and work breakdown structures, and it may be very difficult, if not impossible, to predict how all of the project activities will be organized in advance in terms of Pert charts and Gantt charts. Agile calls for a much more fluid and dynamic approach that is optimized around dealing with a much more unpredictable and uncertain environment.

That doesn't mean that the typical plan-driven approach to project management is obsolete and no longer useful, but it certainly should not be the *only* way to run a project. The challenge for project managers is going beyond a traditional, plan-driven project management approach and learning a much broader range of project management practices in a wider range of environments with potentially much higher levels of uncertainty. It is much more of a multidimensional and adaptive approach to project management.

For example, the person responsible for the success or failure of a project in an agile environment may be called a product owner rather than a project manager, but that person still needs to understand and utilize many principles and practices that have been associated with project management in the past.

It's mostly a matter of applying the project management discipline and skills in a very different context that emphasizes a more adaptive approach to maximizing business value in a very uncertain environment, rather than a more plan-driven approach that attempts to maximize predictability and control.

To understand the potential role of an agile project manager, we need to first get past some of the stereotypes and misconceptions that exist about both traditional project management and agile. The following are some of the stereotypes that exist about project management:

- **Project managers are very command-and-control oriented.**

  Project managers are noted for getting results and sometimes that means being assertive and somewhat directive to set goals and manage the performance of project teams. Many times that behavior is expected of a project manager by the businesses that they operate in. In many companies, if a project team is underperforming, the project manager is the one held responsible and is expected to take corrective action to get the project on track.

- **Project managers are rigid and inflexible and only know how to manage by the waterfall methodology.**

  For many years, project managers have been held accountable for managing the costs and schedules of projects, and we all know that in order to meet cost and schedule goals, you have to control the scope of the project. That, in turn, requires a disciplined approach to defining and documenting detailed requirements and controlling changes, where changes become the exception rather than the norm.

  The emphasis on managing costs and schedules requires accurately defining the requirements up front, which leads to extensive use of plan-driven or waterfall-style methodologies that are based on trying to define the project requirements in detail upfront before the project starts and controlling changes once the project is in progress.

- **Project managers are just not adaptive and cannot adapt to an agile environment.**

  Like the other stereotypes, there may be some amount of truth in this stereotype, but it would be inaccurate to generalize and say that this is true of all project managers. Agile will require some considerable rethinking of the project management approach and some project managers are so heavily engrained in the traditional way of operating because it has been so widely accepted as the norm for such a long time that they may have a difficult time adjusting to an agile project approach.

It should be apparent from the above that project managers are a product of the environment that they work in and what is expected of them. Developing a new image of a project manager may require some changes to the environment that project managers work in and what is expected of them. These are not insignificant challenges, but it certainly isn't impossible for a project manager with a traditional project management background to adapt to an agile environment. A good project manager knows that you have to adapt the project management approach to fit the problem rather than force-fitting every problem or project to a single approach.

AGILE PROJECT MANAGEMENT SHIFTS IN THINKING

The career path for a project manager who has a traditional project management background to become an agile project manager is not easy because it can require some significant shifts in thinking from some of the accepted beliefs that many project managers have had for a long time. Making a shift in thinking is probably one of the most significant changes that a traditional project manager has to make in developing a more adaptive approach to operate in an agile environment.

Emphasis on maximizing value versus control

The first major difference in an agile project management role is the emphasis on maximizing value versus the traditional emphasis on control. Figure 8.1 shows the traditional project management iron triangle, which is based on an emphasis on control that has represented a primary focus of the project management profession for many years:

- The idea of it is to fix the scope of the project and put the primary emphasis on controlling the project cost and schedule by controlling changes to scope. A project was deemed successful if it met the original requirements within the budgeted cost and schedule.
- With those constraints, the three legs of the triangle are set and you can't change any one leg of the triangle without affecting the other legs. For example, for a given scope, you cannot decrease the time of the project without also changing the cost.

This model has been the predominant model in project management for a long time. The problem with this approach is that an excessive emphasis on managing the costs and schedule of a project can lead to a relatively rigid and inflexible style of project management that is not adaptive to uncertain or changing user needs. In many cases, it is just unrealistic to be able to completely define all the requirements for a project in detail. For that reason, many projects have met their cost and schedule goals but failed to deliver the business value required by the user.

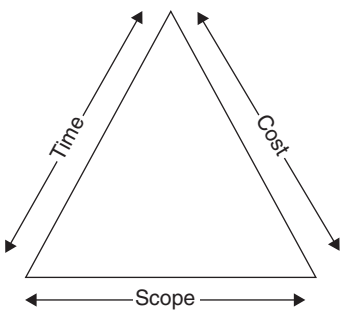


FIGURE 8.1 Traditional project management iron triangle

The project management triangle in an agile project is much more complicated because the constraints may not be as rigidly fixed, change is the norm rather than the exception, and there are many more trade-offs to be made in a much more dynamic environment. So, it really is a challenge to apply some level of project management discipline in that kind of environment and it requires a very different approach, as shown in Figure 8.2.

The impact of putting more emphasis on business value is a significant shift in the focus of decision-making and the primary responsibility for success or failure of the project:

- In a traditional project, the primary decision making is focused on managing costs and schedules, the project manager is one of the primary decision-makers, and has primary responsibility for the success or failure of the project in meeting cost and schedule goals. Since the requirements are assumed to be fixed, the business involvement can be limited once they're signed off, and only significant changes in requirements need to be escalated to the business for approval.
- In an agile project, the primary decision making is focused on maximizing business value, and the product owner, as a representative of the business, is the key decision maker. Since requirements are never completely signed off, the business and the product owner need to be much more continuously involved in the project, and are held responsible for the success or failure of the project in providing the required business value.

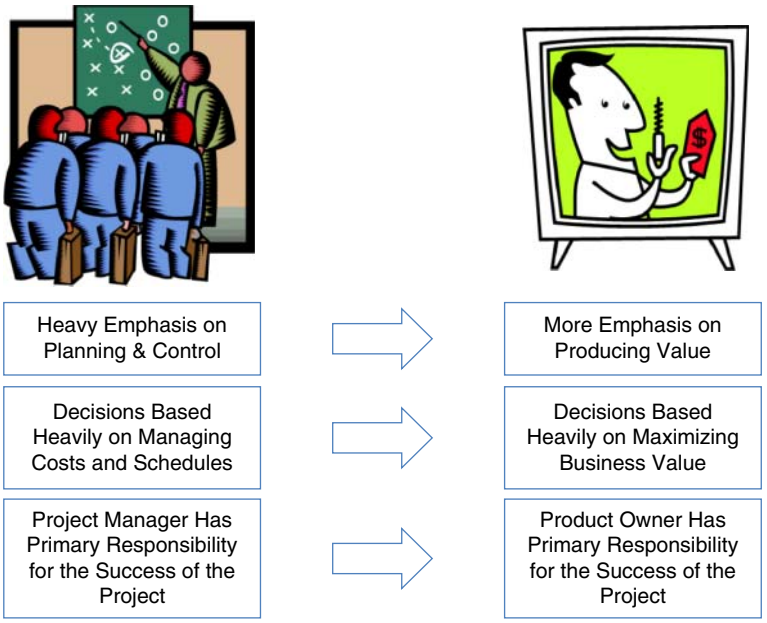


FIGURE 8.2 Value-based project management approach

In an ideal world, as I've mentioned, there is no role for a project manager in a small, single-team agile project, and the product owner takes on most of the responsibilities that would normally be held by a project manager. However, in the real world, in many companies, product owners are not really well-prepared to take on that responsibility, and a project manager will many times play a supporting role to assist the product owner in fulfilling that responsibility.

Project managers are noted for getting results and driving projects to successfully achieve cost and schedule goals, so it does require somewhat of a shift in thinking to focus on delivering value as the primary goal.

Emphasis on empowerment and self-organization

In addition to the shift in decision-making responsibility, there is also a significant shift in the style of how the project is managed to more empowerment of the individuals on the project team to plan and organize their own work without too much detailed direction, as shown in Figure 8.3.

- In a traditional project, the project manager typically might play a very active role in organizing and leading the team to the extent of organizing, planning, and tracking of tasks required to successfully complete the project.
- In an agile project, the team is expected to be self-organizing. The team reaches agreement with the product owner on the goals of each sprint, and takes responsibility for planning and organizing tasks that are needed to complete those goals.

In an ideal world, the Scrum Master takes over the project management function in leading the team, but it is a different kind of leadership; the Scrum Master is regarded as a servant leader; he/she plays a facilitation role to help the team plan and organize the activities rather than being overly directive.

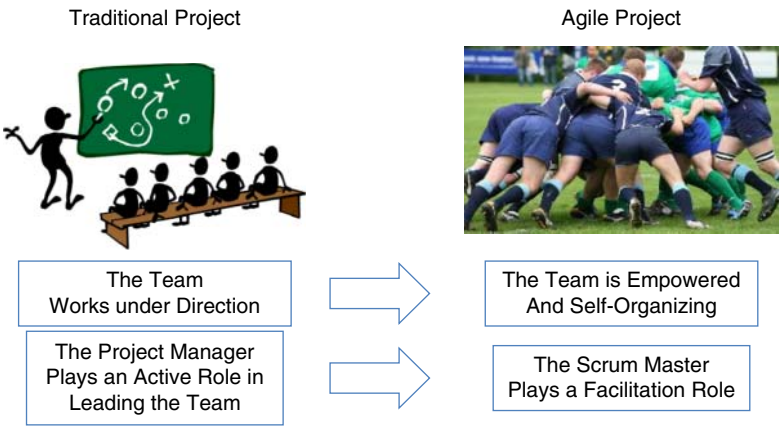


FIGURE 8.3 Impact of empowerment and self-organization

In the real world, many teams have difficulty becoming completely self-organizing and more active leadership is needed to steer them in the right direction. Also, the Scrum Master may or may not have all the project management skills needed to take an active leadership role, help the team organize its activities, and remove obstacles to making the team productive. For that reason, many times companies will combine the role of a Scrum Master with a project manager and call it an agile project manager.

Limited emphasis on documentation

One of the important roles played by a project manager in a traditional project is in developing and managing documentation required by the project because documentation such as requirements documents and specifications are important deliverables in themselves. In fact, in a traditional project, many of the project deliverables are heavily associated with documentation and would not be considered complete without the required documentation.

In an agile project, documentation plays a much more limited role and the emphasis is shifted to producing tangible and demonstrable results with much more limited documentation. As noted in Chapter 3, there is a popular misconception that agile projects do not require documentation at all, but that is not the case. Instead, any documentation should provide value in some way to the project.

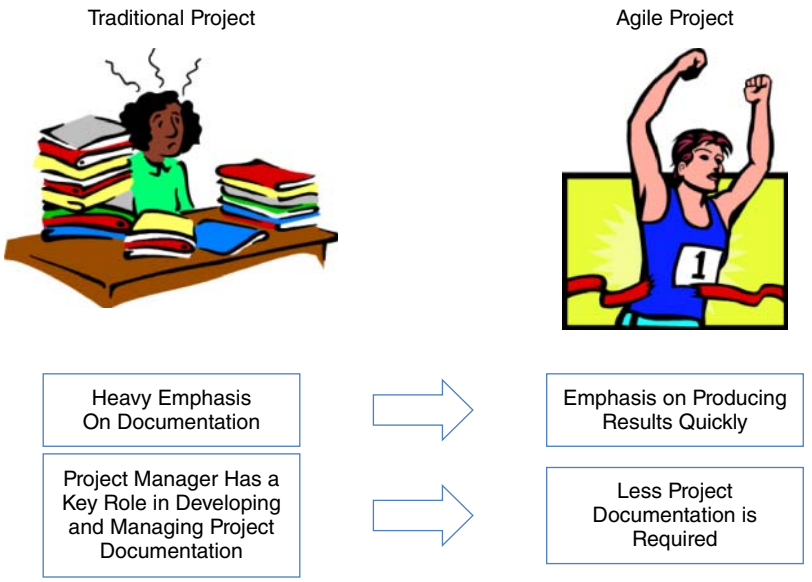


FIGURE 8.4 Limited emphasis on documentation

An important role for a project manager in an agile project is to implement tools that help the team operate more efficiently and many times that will include online electronic tools that take the place of documentation.

Managing flow instead of structure

A very big change in the role of an agile project manager is a shift in emphasis from managing structure to managing flow:

- In a traditional project, project managers are heavily-oriented toward planning and organizing project activities in the form of work breakdown structures, Gantt charts, PERT charts, and others that are all oriented around the structure of the project.
- In an agile project, many of those things become somewhat irrelevant because you may not have sufficient information in the front-end of the project to define the project structure. In an agile project, structure is considerably simplified and is understood to be much more fluid. The role of a project manager shifts to managing flow to optimize the efficiency of the project. That role is somewhat of an agile coach or process engineer role.

  This is also consistent with the focus on helping the team operate most efficiently.

POTENTIAL AGILE PROJECT MANAGEMENT ROLES

The question you might ask at this point is, where does that leave the project manager in a typical agile/Scrum project? If a lot of the planning and decision-making responsibility is shifted to the product owner, there is less emphasis on active team leadership, and less emphasis on documentation; what's left for the project manager to do?

The role of a project manager in an agile project is not well-defined. Officially, there is no role for a project manager at the team level in an agile project. However, there are a number of potential roles that a project manager with the right skills and training to perform the agile project/program manager role can perform.

Making agile work at a team level

In a typical, pure agile project, the tasks associated with making agile work at a team level typically involve training the team members in agile principles and practices and then coaching them in successfully applying those principles and practices in a real world project. (Hybrid agile projects that require blending an agile approach with some level of traditional, plan-driven project management will be discussed later.) Agile relies very heavily on the skill and judgment of the individuals performing the process, and it can take a lot of training, coaching, and mentoring to get the people on an

agile team to a level of full proficiency. Those tasks are many times handled by an agile coach. Rachel Davies, author of *Agile Coaching,* explains what an agile coach does:[1]

> In a nutshell, an agile coach helps teams grow strong in applying Agile practice to their work. It takes time to adopt these changes so you can't do this effectively as a seagull consultant or trainer who swoops in to deliver words of wisdom and then makes a sharp exit. You need to spend time with a team to help them to become more aware of their workflow and how to collaborate effectively.
>
> How is being a coach different from a team lead or project manager job role? Well, it's not incompatible. The difference is that these roles have a wider set of project and company specific responsibilities, such as reporting progress, performance appraisals, etc. I notice that the pressure to deliver can distract from a focus on process improvement. Whereas, if you work solely as an agile coach, you can make this your sole focus because you don't have responsibility for project deliverables and administrivia.
>
> Being a coach is also different because it's a transitory role not tied into project duration. Your goal is for the team to become self-coaching and adept in applying agile then you move on. That doesn't limit agile coaches to introducing agile into organizations and establishing new agile teams. The majority of the teams that I coach are already applying agile techniques and seek coaching because they want to boost their performance and proficiency in agile software development.

Can a project manager add value at that level? It is certainly worthwhile for an agile project manager to have some agile coaching knowledge and skills, but being an agile coach is a fairly specialized role and it takes a level of focus to do it well. An agile project manager should certainly understand the role of an agile coach at least to the extent of learning to recognize performance issues in an agile team that might require the skills of an agile coach; however, it might be very difficult and awkward in many cases for a project manager to try to perform the role of an agile coach.

Although there might be no one with the formal title of "project manager" at the team level in an agile project, a number of project management functions and skills are still needed—they are just distributed among the other roles in an agile project rather than being done by one single person with the title of "project manager". For example, the Scrum Master has responsibility for team leadership and facilitation, and the product owner has overall responsibility for the success of the project and for planning and prioritizing the requirements of the project.

However, that kind of ideal world doesn't really exist in many typical companies, and the people who are in those roles sometimes need help in fulfilling those responsibilities. For that reason, many companies will combine a project management role with one of the other roles on the agile team.

[1]Blog, "Agile Coach: Understanding the Role," posted by Jessica Thornsby, February 25, 2011, http://jaxenter.com/agileagile-coach-understanding-the-role-35038.html.

What I've seen most frequently is companies that combine the Scrum Master and project manager role. That is probably the easiest combination to make. The project manager role could be combined with the product owner role, but that requires a considerable amount of business decision making and business domain knowledge that a typical project manager might not have.

A well-trained agile project manager can help put in place the right people, process and tools in a Scrum project to make the project work most efficiently and will help the team blend the right level of traditional project management with agile principles and practices into their process and roles. However, that is a very different role than a traditional project management role and can require some significantly different skills and training.

Hybrid agile project role

As I've mentioned, I don't believe it is a binary choice between waterfall and agile; there are a number of alternatives between those extremes, as shown in Figure 8.5. For example, many IT services companies do software application development under contract for clients. Naturally, their clients are not going to write a blank check for a software development project without some expectations of the cost, and it would be irresponsible for an IT Services company to commit to a cost estimate without some level of detail of what the requirements are. Of course, the level of those estimates can range from a firm, fixed-price contract at one extreme based on more detailed requirements to a fairly rough budgetary estimate and very high-level requirements at another extreme. The important thing is that both the IT service provider and the customer have a mutual understanding of the level of accuracy in the estimates.

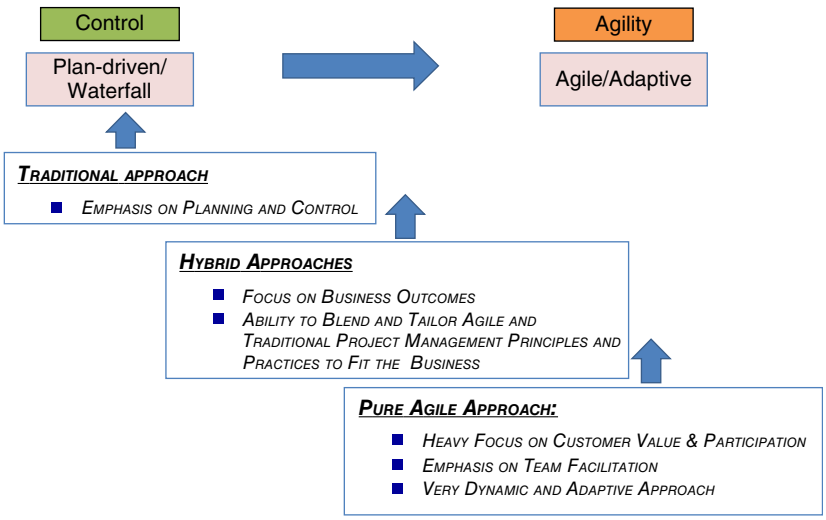


FIGURE 8.5 Hybrid agile approach

This role is a very challenging role for an agile project manager and may involve:

- Blending traditional agile and project management principles and practices in the right proportions to develop a hybrid management approach when required.
- Taking a more iterative and adaptive approach to traditional project management projects.

There are a number of situations where a hybrid agile approach might be needed. Here are a few examples:

1. *Many contracting scenarios require a commitment to some kind of cost and delivery schedule yet still require some flexibility in defining the requirements*. At first, those two things might seem to be mutually incompatible, but they are not—this book provides more detail on a methodology I have used to manage a large government contract, and I will also go over a case study of a defense contractor (General Dynamics, UK Limited) who also successfully developed such an approach.
2. *In fields such as in the medical or pharmaceutical industries, you might need to show traceability over requirements and sufficient control over testing and release processes to ensure that the overall process complies with any regulatory control requirements*. At first, this might also seem to preclude any kind of agile approach; however, that is also not the case—there are ways to intelligently blend agile and traditional project management principles and practices in the right proportions to fit the situation.
3. *As projects are scaled to larger, more complex enterprise-level projects, there is also a need for a significant amount of project management*. Such projects might involve multiple teams and possibly significant levels of coordination with other activities such as training, support, and operational cutover. We will discuss this scenario separately.

Enterprise-level implementation

Above the team level, there is a much broader range of opportunities for a project manager to add value at an enterprise level:

- Managing large, complex enterprise-level projects that require multiple teams and may also require a significant level of coordination and integration with activities outside of the project team
- Aligning an agile development process with a company's overall business objectives and culture and developing an overall strategy that blends agile and traditional project management principles and practices in the right proportions to fit the company's projects and business environment
- Planning and leading agile transformations at the enterprise level

Those tasks are a more logical extension of a project manager's skills and are more likely to utilize the value-added that a project manager can provide.

Let's look at these roles more closely.

Enterprise-level Project/Program Management Role

There is a potential role for an agile project/program manager at the enterprise level in:

- Providing project/program management of large, complex initiatives requiring additional planning and management above the team level and helping to align team-level activities with the company's business goals
- Planning and leading enterprise-level agile transformations designed to align with the company's business objectives, particularly those that require a blended approach of agile and traditional project management principles and practices

Both of these roles will require a number of skills, including program management skills as well as an understanding of how to blend agile and traditional project management practices together at the enterprise level to fit a company's business. At the enterprise-level, the shift in thinking that a project manager needs is similar to the team-level role; however, at the enterprise-level, the role may be more of a program management role and, for that reason, it may require more of a strong active leadership role than just a facilitation role.

An example of such a large-scale enterprise-level project is Harvard Pilgrim Health Care. There is a case study on Harvard Pilgrim Health Care later in this book. The project involved almost completely replacing most of Harvard Pilgrim's legacy systems over a five-year period involving about 100 agile teams.[2]

The project used a hybrid approach based on a combination of traditional, plan-driven project management and an agile development process. It was further complicated by the fact that a large percentage of the development effort was outsourced to another company. Some of the more significant success factors were:

- Developing a very collaborative relationship with the company that was contracted to do the development effort, since most of the development effort was outsourced
- Having an integrated hybrid methodology to manage the effort
- Strong leadership with early planning and well-trained people

The results were very significant. During the five-year duration of this project, which involved a massive redesign of many of Harvard Pilgrim's core business systems, Harvard Pilgrim continued to be named #1 for member satisfaction and quality of care for over nine consecutive years. That's an amazing accomplishment considering the number of moving parts involved in this effort that all had the potential for impacting Harvard Pilgrim's business.

[2]Charles Cobb, *Managed Agile Development—Making Agile Work for Your Business* (Outskirts Press, 2013).

Aligning an Agile Approach with a Company's Business

Agile is not something that should be done mechanically:

- An effective implementation of agile depends on a set of people who have a common understanding of the principles behind agile and who are also driven by a common set of values.
- A consistent set of principles and values is essential to build cross-functional teamwork and cohesion behind everyone involved both within the team and the business sponsors who are directly engaged with the team.

An agile development process is not likely to be as effective if it is done mechanically without a common understanding of the values and principles behind it among both the people on the agile team as well as the people who directly interact with the team or are impacted by the project, such as business sponsors and stakeholders. This often causes some level of conflict if the company's culture is not well-aligned with an agile development process. If there is a misalignment between the values and principles of agile and the culture of the company, it can lead in some cases to trying to overcome those differences.

Changing the company's culture so that it is more conducive to agile is desirable, if that's possible; however, it's often not that simple:

- A company's culture should be shaped around whatever makes sense to drive their primary business and that may or may not be in alignment with an agile development process.
- Changing the culture of a company is not easy to do and is typically not something that can be done quickly.

For those reasons, it is often necessary to find a compromise, at least in the short term, between an ideal world where the company has adopted a complete agile model from top-to-bottom and a more pragmatic approach that recognizes the need to blend an agile development approach with the company's existing culture and business processes. Corporate culture and enterprise-level agile transformations will be further discussed later in this book.

Integration with Enterprise-level Management

Many companies have a well-established existing PMO project management structure and perhaps other levels of enterprise management, such as project/product portfolio management that you just can't unravel overnight. At an enterprise level, many of these companies may have a command-and-control style management approach that would benefit from being replaced with a more sophisticated and more adaptive management approach that is more consistent with an agile development process but it may take time to make that transformation.

The strategy an agile project manager might take for integrating a project management strategy with the company's business may be very different from one company to the next. The challenge for an agile project manager is in helping the company determine how much, if any, of the company's

existing enterprise-level management approach makes sense to keep in place to provide the right balance of control and agility. This decision is very much related to the previous discussion about aligning an agile approach with the company's business.

Aligning an agile project management approach with a company's business and integrating the approach with existing enterprise-level management processes will be discussed in Chapter 12, Scaling Agile to an Enterprise Level, and Chapter 13, Adapting an Agile Approach to Fit a Business.

Using agile concepts in non–agile projects

Even if you never manage a true agile project, knowledge of the agile concepts is likely to significantly enhance the project management skills of someone who only has a traditional project management background for almost any kind of project. Here are a few examples that can go a long way to improve a traditional project without becoming 100 percent agile:

- Developing a more collaborative approach with the business users
- Putting more emphasis on maximizing business value
- Taking a more iterative approach to deliver value incrementally
- Reducing unnecessary documentation and overhead

AGILE AND PMBOK®

Some books have attempted to map PMBOK® to agile principles and practices (Michelle Sliger's and Stacia Broderick's Book, *The Software Manager's Bridge to Agility*[3] is an example). Although you can make some general comparisons, it's like comparing apples and oranges, in my opinion. There may be some similarities, but the whole philosophy behind PMBOK® is very different from an agile approach.

The difference between explicit and tacit knowledge

To better understand this comparison, it is essential to understand the difference between tacit and explicit knowledge. Here's a good definition of these two different kinds of knowledge:[4]

- *Explicit knowledge* is codified knowledge found in documents, databases, etc.
- *Tacit knowledge* is intuitive knowledge and know-how, which is:
  - Rooted in context, experience, practice, and values
  - Hard to communicate—it resides in the mind of the practitioner

[3]Michelle Sliger and Stacia Broderick, *The Software Manager's Bridge to Agility* (Reading, MA: Addison-Wesley, 2008).
[4]"The Different Types of Knowledge," KMT – An Educational KM Site (2013), http://www.knowledge-management-tools.net/different-types-of-knowledge.html

- The best source of long-term competitive advantage and innovation
- Is passed on through socialization, mentoring, etc.—it is not handled well by systems that try to document and codify that knowledge.

This site goes on to define another category of knowledge, called embedded knowledge, which is defined as follows:

- *Embedded knowledge* refers to the knowledge that is locked in processes, products, culture, routines, artifacts, or structures. Knowledge can be embedded formally (e.g., through a management initiative to formalize a certain beneficial routine), or informally as the organization applies the other two knowledge types.[5]

PMBOK® is heavily based on a plan-driven approach—there have been some attempts recently to make PMBOK® more compatible with adaptive project management approaches:

- PMBOK®—Fifth Edition, which has been recently released incorporates some more inclusion of adaptive thinking but it is still heavily oriented around a plan-driven approach.
- PMI® has recently introduced the Software Extensions to the PMBOK® Guide which has a little bit more mention of agile but not much.

However, the fundamental problem is that the whole concept behind PMBOK® is not very compatible with an agile or adaptive approach.

- PMBOK® is clearly based on the idea of explicit knowledge—it is based on the idea that you can codify a checklist of the things you need to do or consider in almost every imaginable project management situation.
- Agile is based much more on the idea of tacit knowledge—instead of attempting to define an explicit checklist of things to do or consider in every possible situation, agile tends to define some very broad-based principles, values, and practices that you need to interpret in the context of the situation you're in.
- Agile also uses the idea of embedded knowledge by embedding knowledge in some widely used agile processes such as Scrum.

These are very different approaches to how knowledge is managed and attempting to modify PMBOK® to make it more agile has some significant limitations for that reason.

Agile is a different way of thinking, in my opinion—it acknowledges that we don't know everything we need to know and puts an emphasis on rapid and adaptive learning. We live in a different world today—in the early chapters of this book, I provided some references on how the pace of adoption of new technology has changed very rapidly. It will probably be difficult for a document like PMBOK® to keep pace with that rate of change. The whole concept behind PMBOK® and the role it plays in defining project management principles and practices may need some significant rethinking, in my

[5]Ibid.

opinion, to fully embrace agile (adaptive) as well as traditional plan-driven approaches to project management.

Relationship to traditional project management functions

In addition to the shifts in thinking previously discussed, the way many standard project management functions are done may be very different. First, some of these functions may be distributed among the various roles on an agile team and may not be done by someone with the title of "project manager". Second, the way these functions are implemented may also be very different.

Each of the major project knowledge areas identified in PMBOK version 5 will be discussed in the following sections. (Sliger and Broderick's book, *The Software Project Manager's Bridge to Agility* is a good resource for more detail.)

It is very important to note that the differences between a plan-driven and an adaptive approach discussed here are meant to be relative. In other words, there is a continuous spectrum of approaches between a totally plan-driven and totally adaptive project rather than a binary choice between two extremes.

Project Integration Management

PMBOK®—Fifth Edition defines *project integration management* as follows:

> Project Integration Management includes the processes and activities to identify, define, combine, unify, and coordinate the various processes and project management activities with the Project Management Process Groups.[6]

The way this effort is described in PMBOK® is very much oriented around a plan-driven project and is also very document-intensive:

- Develop project charter.
- Develop project management plan.
- Direct and manage project work.
- Monitor and control project work.
- Perform integrated change control.
- Close project or phase

[6] *A Guide to the Project Management Book of Knowledge* (PMBOK Guide) 5th ed., (Newtown Square, PA: Project Management Institute, Inc. 2013), p. 63.

Typical Plan-Driven Approach	Typical Adaptive Approach
There is an emphasis on upfront project definition and planning.	This is more of a rolling-wave planning approach.
There is a reliance on documented plans.	The approach is expected to evolve as the project progresses.
The approach is directive for managing and controlling work.	There is more emphasis on providing higher-level direction to self-organized teams.
Emphasizes defining and managing the structure of a project (WBS, Pert charts, Gantt charts, etc.).	Emphasizes managing flow work-in-progress (WIP), Velocity, etc.

At the team-level in an agile project, the product owner has responsibility for defining the level of upfront planning that is appropriate for the project, based on the level of uncertainty in the project and other factors.

Project Scope Management

PMBOK®—Fifth Edition defines *project scope management* as follows:

> Project Scope Management includes the processes required to ensure that the project includes all the work required to complete the project successfully. Managing the scope is primarily concerned with defining and controlling what is and is not included in the project.[7]

The approach for managing project scope in an agile project is completely different. In a traditional project management approach, controlling scope is probably one of the most important project management functions because there is so much emphasis on meeting the project cost and schedule goals. In an environment with high levels of uncertainty, a much more adaptive approach that recognizes the level of uncertainty in the environment is needed. A major strength of agile is that it embraces that change will happen—as a result, an agile project manager and the team come up with ways to deal with changes rather than try to prevent them (traditional change control). That being said, it's worth noting that change isn't constant (i.e., daily), a common misperception of agile. The team does need scope stability for short periods of time (i.e., during a sprint), and change is introduced in between those periods of time.[8]

[7]Ibid., p 105.
[8]Liza Wood, Book Review Comments, May 26, 2014.

Typical Plan-Driven Approach	Typical Adaptive Approach
The approach discourages and controls changes to manage scope in order to manage the project costs and schedules.	Changes are expected and encouraged in order to maximize the value that the project provides to the user.

At the team-level in an agile project, the product owner has responsibility for defining and managing the scope of the project and how rigorous the scope needs to be managed in order to satisfy the business sponsor's expectations.

Project Time Management

PMBOK® defines *project time management* as follows:

> Project Time Management includes the processes required to manage the timely completion of the project.[9]

Typical Plan-Driven Approach	Typical Adaptive Approach
The project approach is based on a "contractual" commitment between the project team and the customer to deliver the specified requirements in an specified amount of time.	The approach is based on a partnership between the project team and the customer to collaboratively make tradeoffs to maximize the value delivered against time required as the project progresses.

At the team level in an agile project, the product owner is expected to continuously prioritize the items to be delivered to maximize the trade-offs of value-delivered versus time as the project progresses.

Project Cost Management

PMBOK® defines *project cost management* as follows:

> Project Cost Management includes the processes involved in planning, estimating, budgeting, financing, funding, managing, and controlling costs so that the project can be completed within the approved budget.[10]

[9]PMBOK Guide, p. 141.
[10]Ibid., p. 193.

Project cost management is similar to time management. In an agile project, the resources are ideally fixed and committed for the duration of the project so the costs are directly related to the time required to complete the project. The product owner controls the costs in the same way he/she controls the overall time required to complete the project.

Project Quality Management

PMBOK® defines *project quality management* as follows:

> Project Quality Management includes the processes and activities of the performing organization that determine quality policies, objectives, and responsibilities so that the project will satisfy the needs for which it was undertaken. Project Quality Management uses policies and procedures to implement within the project's context, the organization's quality management system and, as appropriate, it supports continuous process improvement activities as undertaken on behalf of the performing organization. Project Quality Management works to ensure that the project requirements, including product requirements, are met and validated.[11]

Quality management in an agile project is very different. The following table shows a summary of the differences.

Typical Plan-Driven Approach	Typical Adaptive Approach
Quality testing is typically done sequentially after development is completed.	Quality testing is done concurrently with development.
Quality testing is typically the responsibility of an independent group whose responsibility is to validate the level of quality.	Quality testing is the responsibility of the entire team who produces the product—any quality testers should be an integral part of the project team.
Final acceptance testing typically occurs at the end of the project.	Acceptance testing is done incrementally at the end of each sprint on the items produced in that sprint.

In an agile project, the team is responsible for the quality of the product it produces and the product owner is responsible for performing acceptance testing to validate that it provides the appropriate business value.

[11] Ibid., p 227.

Project Human Resource Management

PMBOK® defines *project human resource management* as follows:

> Project Human Resource Management includes the processes that organize, manage, and lead the project team. The project team is comprised of the people with assigned roles and responsibilities for completing the project. Project team members may have varied skill sets, may be assigned full or part-time, and may be added or removed from the team as the project progresses. Project team members may also be referred to as the project's staff. Although specific roles and responsibilities for the project team members are assigned, the involvement of all team members in project planning and decision-making is beneficial. Participation of team members during planning adds their expertise to the process and strengthens their commitment to the project.[12]

The approach for human resource management in an agile project is also likely to be different. The following table shows a summary of the major differences.

Typical Plan-Driven Approach	Typical Adaptive Approach
Resources are loosely knit and may be brought in-and-out of the project as needed at different times to fulfill commitments.	An agile approach typically is based on a dedicated team of resources who are committed to a project for the duration of the project.
Functional managers typically provide direction to the resources that they are responsible for in a project. A project manager usually provides a coordination function to manage commitments among all the participating groups to meet overall project goals.	The team is expected to be cross-functional and self-organizing without a significant need for functional direction from outside of the team. If a project manager is engaged in the project, he/she should be a strong, cross-functional leader and not just a coordinator.

At the team level in an agile project, the team and the Scrum Master are primarily responsible for human resource management. The product owner may play a role if necessary to approve resource commitments that affect the project deliverables.

[12] Ibid., p. 255.

Project Communications Management

PMBOK® defines *project communications management* as follows:

> Project Communications Management includes the processes that are required to ensure timely and appropriate planning, collection, creation, distribution, storage, retrieval, management, control, monitoring, and the ultimate disposition of project information. Project managers spend most of their time communicating with team members and other project stakeholders, whether they are internal (at all organizational levels) or external to the organization. Effective communication creates a bridge between diverse stakeholders who may have different cultural and organizational backgrounds, different levels of expertise, and different perspectives and interests, which impact or have an influence upon the project execution or outcome.[13]

The approach for communications management in an agile project is also very different. The table below shows a summary of the major differences.

Typical Plan-Driven Approach	Typical Adaptive Approach
A significant amount of communications to groups outside of the project team is controlled and funneled through the project manager	There is an emphasis on openness and transparency information is much more freely shared without a project manager becoming an intermediary and a bottleneck in releasing information
Bad news may be guarded and provided to people outside of the project team on a need-to-know basis	All information (good and bad) should be disclosed in the spirit of partnership, openness, and transparency

At the team level in an agile project, the responsibility for communications management is distributed:

- Everyone on the team has a responsibility to communicate the status of their assigned work
- The Scrum Master generally has the responsibility for facilitating the review and discussion of that information in Daily Standups and other forums
- The product owner is expected to be the voice of the project to the business sponsor and stakeholders but he may be assisted in that role by a Scrum Master

[13] Ibid., p. 287.

Project Risk Management

PMBOK® defines *project risk management* as follows:

> Project Risk Management includes the processes of conducting risk management planning, identification, analysis, response planning, and controlling risk on a project. The objectives of project risk management are to increase the likelihood and impact of positive events, and decrease the likelihood of negative events on the project.[14]

The approach for managing risks in an agile project is very different. The following table shows a summary of the differences.

Typical Plan-Driven Approach	Typical Adaptive Approach
In both a plan-driven and adaptive approach, risks are generally related to uncertainties in the requirements	
In order to provide some level of predictability over project costs and schedules, traditional plan-driven projects tend to be very risk-averse and treat risk as something that has to be controlled and avoided.	In order to provide a level of adaptivity to user needs, an adaptive approach recognizes that it may be necessary to live with uncertainty that will be resolved as the project progresses rather than attempting to resolve it all upfront.
A traditional plan-driven project many times attempts to reduce uncertainty to a minimum in order to minimize risks.	As a result, it may be necessary to take a more aggressive approach to managing risks and defer the identification and resolution of some risks until the project is in progress.

The key difference is that an agile approach recognizes that the risk of failing to meet customer requirements in a very uncertain and dynamic environment is very significant and requires an adaptive approach towards managing uncertainty in the project.

> Risks and benefits always go hand in hand. The reason that a project is full of risk is that it leads you into uncharted waters. Projects with no real risks are losers. They are almost devoid of benefit; that's why they weren't done years ago.[15]

Instead of attempting to identify and resolve all major sources of uncertainty and risk prior to the start of the project, an agile project is based on continuously doing risk planning and management

[14]Ibid., p. 309.
[15]Tom DeMarco and Timothy Lister, *Waltzing with Bears* (New York: Dorset House, 2003).

throughout the project. In an agile project, risks can be identified and mitigated at any time, either up front prior to the start of the project or once the project is in progress.

At the team level in an agile project, the responsibility for managing risk is distributed:

- The product owner is expected to be the primary focus for decision-making on risk management.
- However, the entire team facilitated by the Scrum Master is expected to participate in identifying and mitigating risks.

Agile continuously identifies and resolves risks throughout the project (the power of the stand-up and the retrospective) rather than attempting to identify and control them upfront and treat it as a separate activity. That being said, in an agile project it is useful to identify and stay on top of some high-level risks and opportunities as part of the project planning and release planning processes.[16]

Project Procurement Management

PMBOK® defines *project procurement management* as follows:

> Project Procurement Management includes the processes necessary to purchase or acquire products, services, or results, needed from outside the project team. The organization can be either the buyer or seller of the products, services, or results of a project.[17]

The approach for managing procurement in an agile project may also be very different. The following table shows a summary of the differences.

Typical Plan-Driven Approach	Typical Adaptive Approach
Procurement is typically based on having very well-defined requirements and specifications that must be met by the vendor delivering the products or services.	There may be more of a spirit of partnership with the vendor who is delivering the products and services with some level of flexibility to work out details of requirements and costs and schedules
The relationship between the customer and the vendor is typically based on a very well-defined contract, and it may be competitively bid among multiple vendors to get the lowest price.	The contract may be high level, with some latitude provided for working out details and the vendors selected to participate in the contract may be limited to enable more of a partnership relationship.

[16] Wood.
[17] PMBOK Guide, p. 355.

At the team level in an agile project, the responsibility for defining and negotiating the procurement approach will probably belong to the product owner, who may be assisted in that role by a project manager, depending on the size and complexity of the procurement.

Project Stakeholder Management

The approach for managing stakeholders in an agile project may also be somewhat different. The following table shows a summary of the differences.

Typical Plan-Driven Approach	Typical Adaptive Approach
The project manager, supported by a business analyst, if necessary, is responsible for engaging stakeholders in the project and ensuring that their interests are adequately represented.	The product owner is responsible for representing the interests of all stakeholders in the project. He/she may be assisted in that role by a business analyst.
Requirements of the stakeholders are documented and consolidated and approved by the business sponsor as necessary.	There is a much higher level of reliance on direct face-to-face communications in the project.
The project manager plays a coordination role in gathering and consolidating the requirements for approval by the business sponsor as necessary.	The product owner is a decision maker and should be able to approve stakeholder requirements within the scope of his/her responsibility. (He/she may escalate decisions to a business sponsor as necessary.)

At the team level in an agile project, the product owner is expected to be the focal point for managing communications with stakeholders; however, everyone on the team is expected to communicate directly with stakeholders as needed as the project progresses.

SUMMARY OF KEY POINTS

1. Agile Project Management Shifts in Thinking

   There are several significant shifts in thinking that someone with a traditional project management background may have to make to develop the more adaptive approach that is needed in an agile environment:

   - Emphasis on maximizing value versus control
   - Emphasis on empowerment and self-organization
   - Limited emphasis on documentation
   - Managed flow instead of structure

2. Potential Agile Project Management Roles

The role of a project manager in an agile project is not well-defined; and officially, there is no role for a project manager at the team level in an agile project. However, there are a number of potential roles that a project management with the right skills and training to perform the agile project/program manager role can perform. These roles include:

- Making agile work on a team level
- Managing hybrid agile projects
- Enterprise-level project/program management role

However, even if a project manager is not actually involved in a pure agile project, many of the general concepts and principles behind agile are applicable, to some extent, to almost any project. For that reason, learning these principles and developing a more adaptive approach is probably of benefit for any project manager.

3. Agile and PMBOK®

PMBOK® is heavily written around a plan-driven, document centric approach. The philosophy behind PMBOK® is also very different from an agile approach—PMBOK® is more oriented to an explicit knowledge approach—it is based on the idea that you can codify a checklist of the things you need to do or consider in almost every imaginable project management situation. An agile approach is more oriented to a tacit knowledge approach. Instead of attempting to define an explicit checklist of things to do or consider in every possible situation, agile tends to define some very broad-based principles, values, and practices that you need to interpret in the context of the situation you're in.

Comparing PMBOK® and agile is like comparing apples and oranges, in my opinion, because there are so many differences in the fundamental philosophies between the two approaches. However, it is useful to make some general comparisons of the knowledge areas in PMBOK® to understand how the differences with a plan-driven and adaptive approach.

DISCUSSION TOPICS

Project Management Shifts in Thinking

1. What do you think is the most important shift in thinking that a project manager needs to make to operate in an agile environment?

2. What would be the most difficult change? Why?

Potential Agile Project Management Roles

3. What do you think is the most likely role for a project manager to play in an agile project?

Agile and PMBOK®

4. Discuss how you might take a different approach to a project based on a broader understanding of the knowledge areas in PMBOK®.

9 Agile Communications and Tools

AGILE COMMUNICATIONS PRACTICES

ONE OF THE MOST IMPORTANT values behind agile is *openness and transparency* (see Chapter 3). Many traditional projects in the past limited the sharing of information to carefully controlled channels of communication and bad news was sometimes hidden from view in order to present a favorable image of progress. An important role of a traditional project manager in that environment has been to manage that flow of information.

Since the flow of information in an agile project is more open and transparent, information can be allowed to flow much more easily and automatically using the concept of an *information radiator*. For that reason, the role of a traditional project manager to manage the flow of that information is less essential. In addition to the general importance of openness and transparency, there are a number of other reasons why communications is extremely important in an agile project:

- Information is rapidly and dynamically changing in real-time throughout the project and needs to be shared efficiently.
- Good communications is essential to support close collaboration among everyone on the project team as well as people who may be peripheral to the project team, especially if the team cannot be collocated—that is, working in the same area—which is often the case.
- Sharing of information with the customer and business sponsor is also essential for the same reasons to support a close and collaborative partnership relationship.

As agile projects become larger and more complex, using tools to help distribute information quickly and efficiently becomes essential.

Information radiators

An *information radiator* is a large and highly visible display of critical team information that is typically located in a spot where the team and others can see it constantly, and it is continuously updated

either during the daily standup meetings or even more frequently in real-time as work is completed. It could take on several different forms:

- In some cases, it is actually a large white board or a complete wall of a room that is used to track progress.
- It could also be in electronic form by using one of a number of different agile project management tools so that a much broader audience can view the information online.

The concept of an *information radiator* in agile is consistent with the value of openness and transparency:

- Everyone on the team is aware of the work of all other members of the team and how it contributes to the goals of the team, which is consistent with promoting a strong and unified teamwork approach.
- The product owner and other stakeholders outside of the team are also aware of progress and issues that might impact progress which is consistent with promoting a spirit of partnership and customer collaboration.
- A white board approach using a physical white board and colored "stickie" notes has been widely used in agile projects for a long time, but it has some significant limitations.
  - It doesn't work well with distributed teams.
  - It doesn't provide an ability to easily roll up information across multiple teams.
  - It is difficult to keep the information organized and it also doesn't provide a capability to sort and report on the information in different ways.

For those reasons, many agile projects are moving to more widespread use of online tools. Figure 9.1 shows an example of an information radiator using an online tool.

A big advantage of information radiators is that each individual on the project team can individually update the status of tasks that he/she is responsible for in near real-time and all of that information will be aggregated in the information radiator for sharing with others both inside and outside the project team. That reduces the role that a project manager has provided of aggregating and reporting project status information. An information radiator is typically used in conjunction with the daily standup meeting to discuss and review the work being done by the team. If an online tool is used, the information can also be rolled up at a number of different levels across multiple teams and even across multiple projects.

The use of information radiators is a very powerful aspect of agile, and if it is implemented correctly:

- It promotes openness and transparency which helps build stronger and more effective teamwork.
- It promotes customer collaboration and helps to build a spirit of trust and partnership between the development organization and the business users.

- It can remove a huge burden on management for attempting to control all aspects of an operation. In one situation, a senior manager commented on how much agile has removed the management team from resolving day-to-day issues that the project team now takes responsibility for, and it has enabled the company to focus on much higher-level strategic goals.

Face-to-face communications

Agile heavily emphasizes face-to-face communications wherever possible over other forms of communication.

> "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."[2]

However, that does not mean that documentation and other forms of communication are eliminated and this principle needs to be adapted to fit the situation, particularly in the case of widely

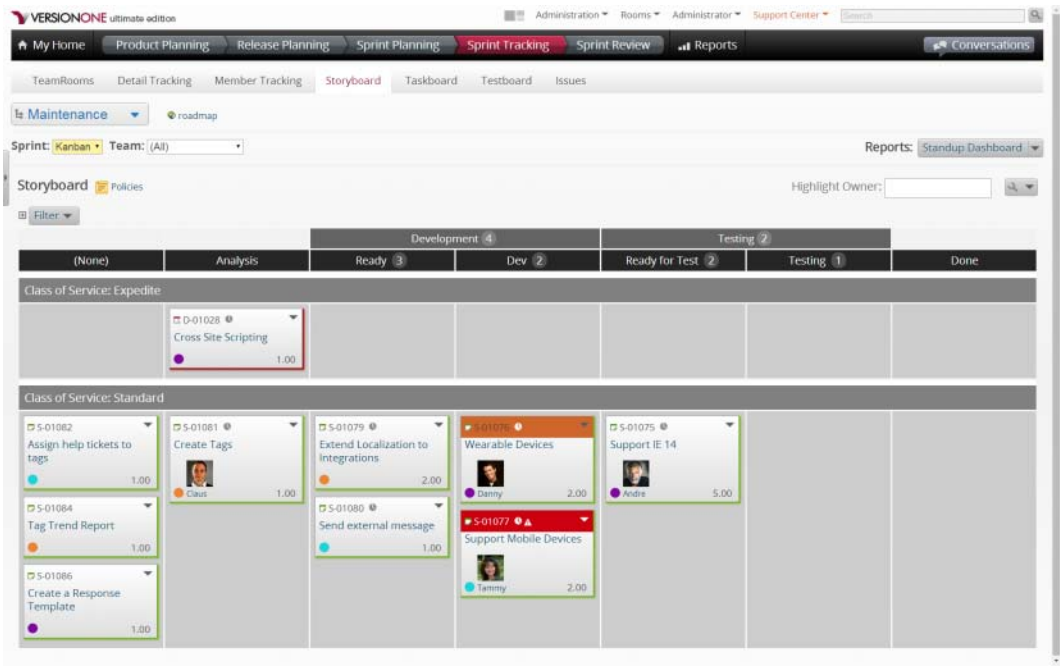


FIGURE 9.1 Example online information radiator[1]

[1] VersionOne agile Tracking Tools, http://www.versionone.com/product/agile-tracking-tool/
[2] "Principles behind the Agile Manifesto," http://www.agilemanifesto.org/principles.html.

distributed teams. Here are some examples of how face-to-face communications are used in agile projects:

- Ideally, agile teams are co-located in the same room to facilitate direct face-to-face communications
- User stories do not attempt to capture all possible details of customer requirements—they are considered to be a "placeholder for communications."

Daily standups

All of us have participated in long, inefficient meetings that are not focused, go on and on, and don't use people's time very well. Daily standups are a way of making meetings much more efficient in an agile project.

- They take place daily but they are limited to 15 minutes, and each person in the meeting typically stands up to encourage people to keep it brief.
- The meeting is focused on the tasks at hand and is typically held in front of a progress board.
- The meetings are structured so that each person answers three primary questions:
  - What did you accomplish yesterday?
  - What are you going to accomplish today?
  - What obstacles are in your way?

Naturally, the idea of a daily standup might need to be modified to fit the environment, particularly with distributed teams.

Distributed teams

Many agile communications practices (e.g., daily standups) are based on the assumption that the team is collocated. However, in many situations, it isn't possible to have that level of collocation and the communications practices need to be adjusted to support distributed teams that may not be in the same geographical location and could even be in different time zones, which is often the case with offshore development teams. In that case, you have to adjust the communications strategy to fit distributed teams:

- Video conferencing might be used in lieu of face-to-face communications.
- Online status boards might be used in lieu of physical wall boards.
- Wikis might be used for sharing of information.
- Daily standups might need to be extended or supplemented with other meetings to provide more communications.

Managing communications with offshore teams in an agile project can be very challenging. Some of the challenges are:

- Geographic separation
- Cultural and language differences
- Time zone differences

I was in a situation where the entire development team was in India and only the customer-facing people were in the United States. Informal meetings next to the water cooler will not work in this situation. But even in an agile environment where the team is collocated, a brief daily standup might not be sufficient. If it is the only (or primary) form of communication with the team throughout the day, it can be very difficult to limit the daily standup to just a 15-minute session. You must learn to adapt the communication practices to the situation.

AGILE PROJECT MANAGEMENT TOOLS

In my opinion, an agile project manager who doesn't know how to use one of the widely used agile project management tools like VersionOne, Rally, Jira, or others is equivalent to a traditional project manager who doesn't know how to use Microsoft Project. The Agile Manifesto value says:

> "Individuals and interactions over processes and tools."

Tools, however, still play a very important role in agile, especially as you start to scale projects to large, complex enterprise-level solutions and that's the area where you are most likely to find an agile project management role.

There are many tools that are used in agile ranging from very simple tools that are oriented around simple, team-based activities to more complete tools that include team-level capabilities but also go well beyond that level and provide a capability for scaling projects to enterprise-level projects. It is impossible to cover all of the agile tools that someone might consider using in this book—in order to bound the scope of the discussion and simplify it a bit, I'm going to do the following:

- First, since this book is about agile project management, I'm going to focus the discussion on tools that are most likely to be used by an agile project manager that would provide a full capability for handling large, complex, enterprise-level projects. Several that I've worked with, in particular, include VersionOne, Rally, Jira/Greenhopper, and Microsoft TFS.
- Since many of these tools have similar capabilities, I've decided to focus on using VersionOne Ultimate Edition (Version 14) as a representative example of all of them to provide a general understanding of how these tools are typically used. This is not intended to be an endorsement of the VersionOne tool; I just had to pick one tool to use as an example to simplify this topic. It is also not

intended to be an exhaustive tutorial on how to use the tool. Most tools such as VersionOne have online videos for that purpose.

Agile project management tools are fundamentally different from traditional project management tools like Microsoft Project:

- The typical way of modeling traditional projects is based on a fairly statically defined model composed of work breakdown structures, Pert charts, Gantt charts, and others. Those modeling tools work fine for a heavily plan-driven approach where the requirements and the overall design approach for meeting those requirements can be defined prior to the start of the project; however, they can be difficult and impractical to apply to a more dynamic and adaptive project approach where the requirements are much more uncertain and difficult to totally define upfront.
- A more dynamic and adaptive approach is based on the concept of a continuous flow of project requirements that are expected to change and be further defined as the project progresses and calls for a different modeling approach. An agile project manager needs to go beyond understanding how to develop a fairly statically defined project model based on a traditional project modeling approach and understand how to optimize the flow of a much more dynamic stream of changing requirements through a much more flexible and adaptive process model. That's what agile project management tools are designed to do.

The fundamental difference is that traditional tools like Microsoft Project are heavily oriented around defining and managing the *structure* of project activities while agile tools are more oriented around managing the *flow* of project activities through a much simpler and fluid structure.

Benefits of agile project management tools

At an enterprise level, agile project management tools provide some significant value:

1. *Ability to fully engage all members of the team in the process.* An agile project management approach is very different from a traditional, plan-driven project management approach. In a traditional, plan-driven project, the project manager is primarily responsible for planning the project as well as managing and reporting progress. Microsoft Project is well-designed to support that role because it is a stand-alone desktop tool and most of the information flows through the project manager. In an agile project, the responsibility for planning and managing the effort is distributed among everyone on the team, and agile project management tools are designed with that in mind.
2. *Ability to update progress in real-time and rapidly view status and issues (information radiators).* An agile project is also fast moving, and it's very important to be able to update information easily in real-time and for anyone to be able to view that information easily. That's the concept of an information radiator.
3. *Ability to scale projects to an enterprise level and provide a standardized way of reporting across projects.* As I've previously discussed, there are different kinds of information radiators.

Small, single-team agile projects might use a simple Kanban board on a white board with index cards or "stickies" on it to manage and track progress. That method works OK for small teams, but falls apart quickly for projects with teams that are not co-located or projects that require multiple teams. Most of the agile project management tools provide at least the capability for an online Kanban board to enable sharing of information across distributed teams.

Characteristics of enterprise-level agile project management tools

VersionOne has developed a nice summary of criteria for selecting a robust, enterprise-level agile project management tool, which is reproduced here with permission from VersionOne.[3]

Full Agile Process Lifecycle Coverage

"Agile management systems must span the full range of processes from integrated customer feedback through portfolio/product planning, code integration, testing and delivery.

- Product, release and iteration planning & tracking
- Agile portfolio management including epic boards, epic ranking, rapid epic breakdown and reporting
- Strategic goals, functional rollups (themes), goal assignment, impediment tracking and retrospectives
- Consolidated requirement backlog and defect repository
- Test management (manage, track and execute tests across multiple stories, defects and distributed projects)
- Integrated product road-mapping linked to releases/iterations

Team and Customer Collaboration

Collaboration capabilities should promote teamwork and expedite communication between team members, teams and organizations.

Social-media style communication portal for project teams

Contextual collaboration includes links to work items, team members and dedicated TeamRooms™

[3] "VersionOne Tool Evaluator Guide," http://www.versionone.com/pdf/agiletoolevaluator.pdf.

RSS feeds and email notifications for receiving message and alerts

Cross-project planning, tracking & reporting for distributed team members

Board Views are customizable, filterable and support flexible color coding to convey information

Integrated customer Idea Management platform (facilitating customer engagement, collaboration, and prioritization)

Visibility, Reporting & Analytics

Agile management solutions must include concise views into data, work estimates, actuals and trends.

- Executive-level dashboards with best practice metrics
- Advanced planning including: release forecasting, "what-if" analysis, and workload balancing
- Team burn-down, burn-up, cumulative flow, and test trend reporting
- Epic boards/Storyboards with drag-and-drop tracking of story cards, work-in-process limits, aging & cycle-time reporting
- Roll-up reporting for teams working across projects and projects with multiple teams
- Best-practice agile dashboards: Project, Sprint, Program, Team Member, and role-based dashboards
- Custom reports and graphs created via web-based, wizard-driven interface
- Agile visualizations—Epic bubble charts, Hierarchy/Tree charts, Relationship Visibility, Release Dependency Mapping

Simplicity & Ease of Use

Use a tool with a simple user interface and built-in process navigation.

- Built-in agile process navigation with customizable dashboards for team members to track their projects and work
- Drag-and-Drop ranking and whiteboard-style release and iteration planning
- Drag-and-Drop epic, story, task and test boards with customizable workflow processes and configurable card data
- Multi-select options for actions such as: Move; Close; Reopen; Delete; and Rank

- Multi-level estimation at the story/defect level and the effort tracking level
- TeamRoom™—dedicated team-based environment supporting the daily activities of development teams
- PlanningRoom™—dedicated environment for program-level managers to collaborate in focused planning sessions

Configurable Workspaces, Process and Terminology

Agile Tools should guide you through the agile process and help you implement "agile your way". Custom workspaces and terminology support unique process configurations without sacrificing visibility, reporting and analytics across the portfolio.

- Drag-and-drop epic, story, task and test boards with customizable workflow and configurable card data
- Customizable methodology templates (XP, Scrum, DSDM, Kanban, etc.)
- Extensive customization options including boards, fields, lists, value, grids, etc.
- Customizable folder structure for nested project/release hierarchy
- Customizable and filterable board views that support color-coded visual indicators

Agile Portfolio and Program Management

Agile tools should grow with you as your needs grow including portfolio-level planning for your strategic initiatives, program-level coordination and project-level story delivery with full traceability from high-level epics to development-level tasks.

- Portfolio-level business initiatives, release rollouts, progress and organizational velocity mapped against a timeline
- Program-level Epicboards, Epic Bubble charts, epic ranking, planning and roll-up reporting
- Integrated product roadmapping linked to releases/iterations
- Advanced planning including: release forecasting, "what-if" analysis, and workload balancing
- Cross-project team planning, tracking and reporting
- Release Dependency Diagram to better prioritize story completion
- Extensive support for Scaled Agile Framework® (SAFe™) PPM methodology (Planning, Process, Metrics, Reports, etc.)

Deployment, Security and Integrations

Agile tools should provide: flexible deployment options for ANY team size and ANY agile methodology; application-, role- and project-level security; and open integrations for simplified deployment and customization.

- Free trial software available for both On-Demand and On-Site options
- Four right-sized product editions that grow with you as your agile project management needs evolve
- SaaS and on premise deployment options with easy data portability should your requirements change
- Available integrations to 45+ ALM technologies
- JAVA and .NET SDKs and open, web-services API
- Password authentication and single Sign-On (SSO) across systems and third-party integrations
- Project-level and role-based security to provide the right access to the right data

That's a long list of capabilities and, naturally not all of those capabilities may be required in a particular implementation, but it's a good list, and it's always easier to take away something that's not needed rather than adding something that's not there.

SUMMARY OF KEY POINTS

Agile Communications Practices

1. Communications in an agile project is extremely important to support openness and transparency and it is essential to support teamwork and rapid and efficient coordination among the people on the project team, as well as close collaboration and partnership with the customer and business sponsor.
2. Information radiators are widely used in agile projects to disseminate information rapidly and efficiently in a very dynamic and fast-paced environment. The simplest form of information radiator is a white board with colored "stickies" to track progress of work. Online tools have a number of advantages over white boards and can be more effective in many environments for a number of reasons such as the ability to easily roll up information across multiple teams and the ability to support distributed teams that cannot be collocated.
3. Agile emphasizes face-to-face communications; however, in the real world, communications practices frequently need to be adapted to fit teams that cannot be collocated. In that situation,

there are a number of alternatives that can be used in lieu of or to supplement direct face-to-face communications. Those alternatives include video conferencing, online status boards, and Wikis.

Agile Project Management Tools

1. Agile project management tools have a very different orientation than traditional project management tools. Traditional project management tools are heavily-oriented around planning and managing the structural aspects of a project (Gantt charts, Pert Charts, dependencies, etc.). Agile project management tools are much more organized around planning and managing flow and the traditional emphasis on structure is considerably simplified or not needed at all.
2. Agile project management tools are also designed around a collaborative team approach where each individual on the team has direct access to the tool for planning and tracking their own work rather than all work being coordinated and managed by a project manager.
3. Many agile project management tools are very scalable and offer capabilities all the way from simple single-team agile projects to complex, large-scale enterprise-level capabilities.

DISCUSSION TOPICS

Agile Communications Practices

1. What are the major differences between communications practices in an agile project and a conventional, non–agile project? Why is communications so important in an agile project?
2. What is an information radiator? What are the advantages of online tools for managing communications in an agile project?
3. What are some of the challenges associated with distributed and/or offshore teams, and how would you go about resolving them?

Agile Project Management Tools

4. What agile tool capability do you think is most important in a typical agile project? Why?
5. What do you think is the most important factor in choosing an appropriate tool for an agile project environment? Why?

10 VersionOne Tool Overview

I'VE CHOSEN TO USE THE VersionOne tool as a representative example of an enterprise-level agile project management tool. This is not intended to be an endorsement of the VersionOne tool—I had to choose one tool to simplify this discussion and I chose VersionOne, just as some traditional project management textbooks have chosen to use Microsoft Project as a representative example of traditional project management tools. I chose VersionOne because it has a relatively complete set of the most important capabilities that I believe an agile project manager would need, it can be used with a variety of widely used development environments, and it is relatively easy to learn and use quickly.

However, the selection of a tool can be a difficult choice—for example, there is a significant trend to provide an overall Application Lifecycle Management (ALM) tool that integrates all aspects of a project including development and test with a project management tool. Microsoft Team Foundation Server (TFS) is an example of a more complete ALM tool that provides a single, well-integrated environment for many aspects of software development, including source code control, testing, and project management.

The following sections provide a walk-through of how a tool like the VersionOne tool could be used to support all aspects of a large scale agile project, including product/project planning, release planning, sprint planning, and sprint tracking. The VersionOne tool also provides a number of higher-level capabilities, including portfolio management, program management, and product road-mapping that are outside of the scope of this book.

PRODUCT/PROJECT PLANNING

Product/project planning is not well-defined in agile, and, for that reason, some people might assume that it is just not done at all. That is not correct—as I've mentioned in a previous chapter, agile

planning can take on a number of different forms but you will rarely find an agile project that starts by writing code with little or no planning whatsoever:

- At one extreme, the product/project planning level might be somewhat open-ended and fairly high-level, and might not include an effort to try to pin down the costs and schedule for the project.
- At the other extreme, there may be a need for a more robust level of planning to more accurately pin down the scope of the project and come up with some kind of estimate of the costs and schedule for the project.

There is no right or wrong answer for the correct level of product/project planning—it is completely dependent on the level of uncertainty in the project and other factors. It is also important to recognize that product/project planning is not done once at the beginning of the project and then put on the shelf. It is typically an ongoing process that is revisited throughout the project. For example, even a robust level of upfront product/project level planning is not likely to be 100 percent complete and will need to be revised and updated throughout the project as new information becomes known.

The VersionOne tool is representative of most agile project management tools in this area—it doesn't provide a prescriptive way of doing upfront planning and is flexible and adaptive to be used with different levels of planning. In this discussion, I will describe how you might use a tool like VersionOne to do a fairly robust form of product/project planning because it is easier to scale down and eliminate what you don't need for a less robust level of product/project planning.

Figure 10.1 shows an example of how I used the VersionOne tool to plan the development of an agile project management course based on this book. I laid out epics of the high-level areas to include in the course, then broke those epics down into further detail and estimated the level of effort associated with each.

The VersionOne tool provides a complete capability for doing product/project planning that includes a centralized view of the product backlog and a drag-and-drop interface to quickly prioritize user stories and align project efforts with business needs using epics, themes, and goals.[1] In addition to the standard product/project planning capabilities discussed in this section, the VersionOne tool also includes some advanced capabilities:

- Portfolio management
- Program management
- Product road-mapping

These won't be discussed here.

[1] "Agile Product Planning," VersionOne, http://versionone.com/product/agile-planning-software/.

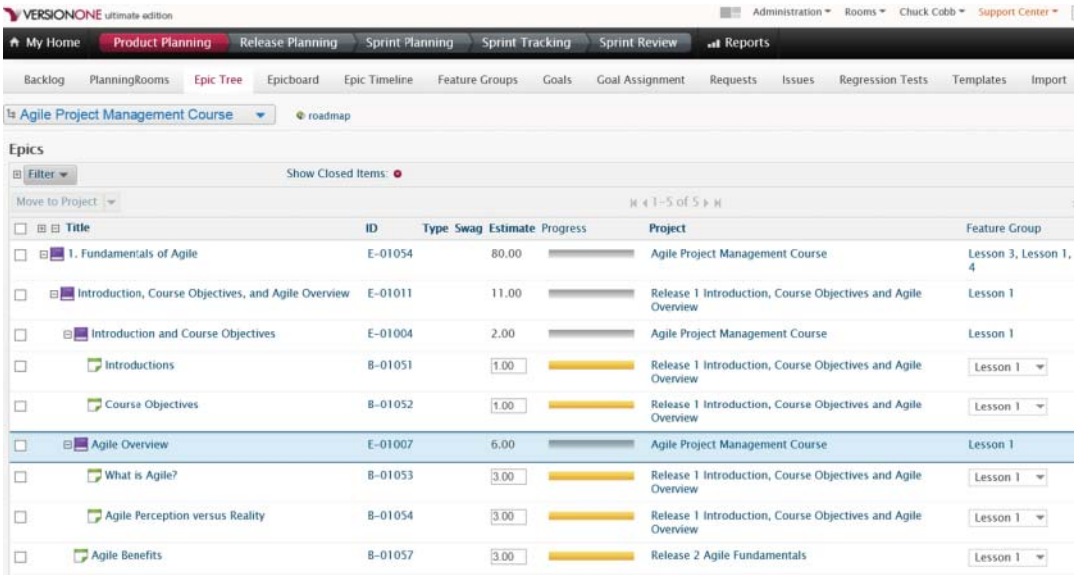


FIGURE 10.1 Example product planning with VersionOne

Product backlog management

An agile project approach typically starts with some kind of product backlog; however, at the product/project planning level, the level of completeness and detail in the product backlog might vary significantly from one project to the next. You might start with a very high-level set of product backlog items and progressively refine it and add further details later in the project or you might choose to go beyond that and do more upfront planning early in the project depending on the level of uncertainty in the project and other factors.

That is exactly the approach I used in designing the agile project management course based on this book. The VersionOne tool provides an online, centralized repository that lets you quickly prioritize stories and epics using drag-and-drop ranking to quickly construct an initial product backlog to whatever level of detail is appropriate. It can be used at various levels of complexity and can make it easier for product owners to manage large, complex backlogs that might also require coordinating and centralizing multiple backlogs in one place using VersionOne's flexible project hierarchy.

The first step in developing an agile project management course based on this book was to develop a high-level list of epics, as shown in Figure 10.2.

The next step was to add more details to each major epic, as shown in Figure 10.3.

For each story, the next step was to add tasks to indicate the work to be done on each story. Opening each story allows adding more detail to the story including tasks, test cases, dependencies, and issues, as shown in Figure 10.4.

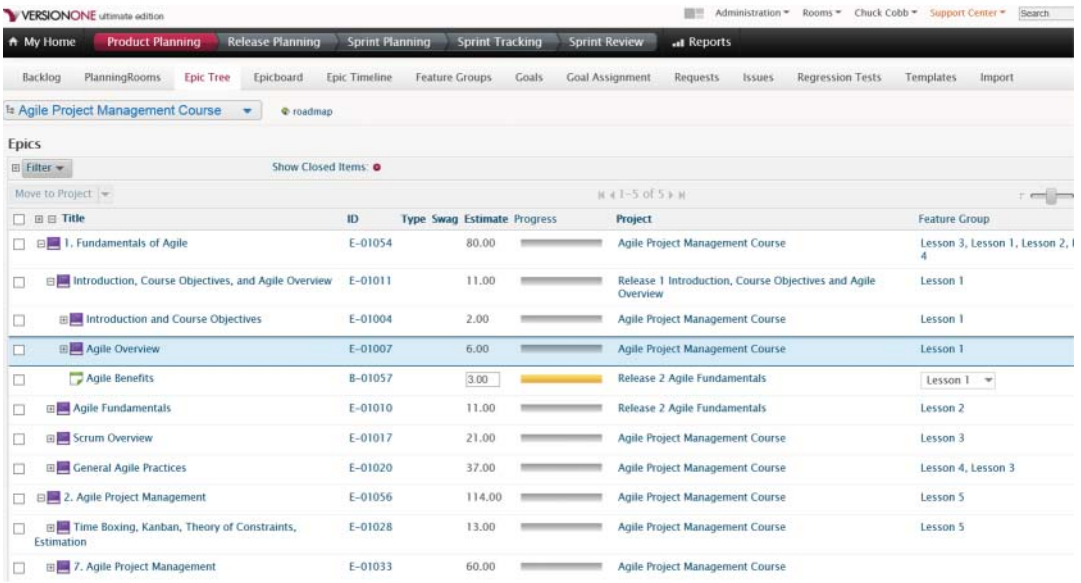


FIGURE 10.2 Example of high-level epics with VersionOne

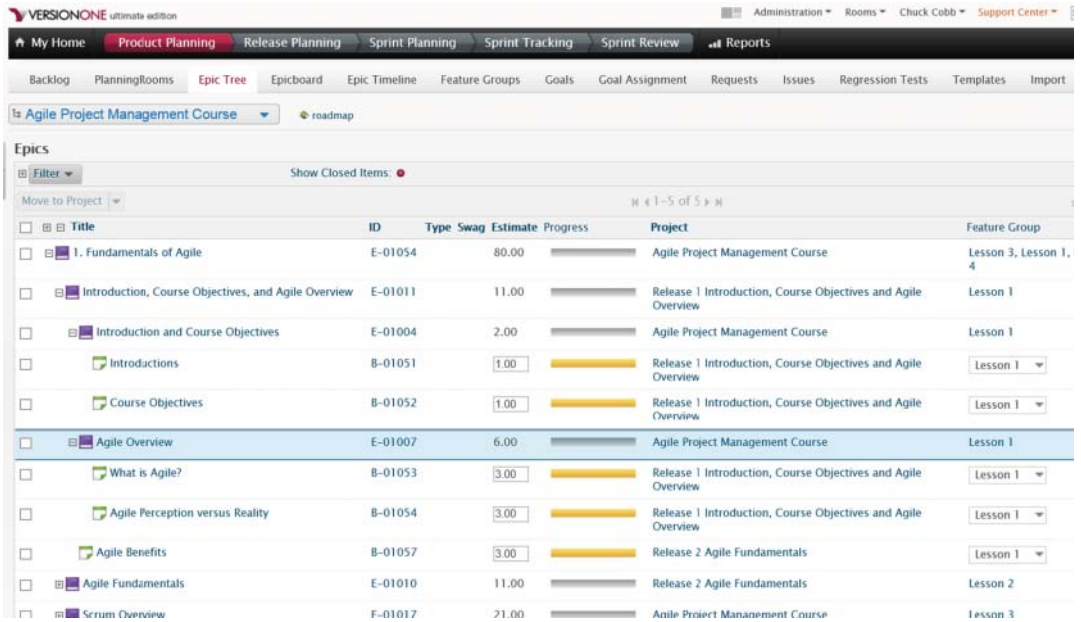


FIGURE 10.3 Example of adding more detail to VersionOne product backlog

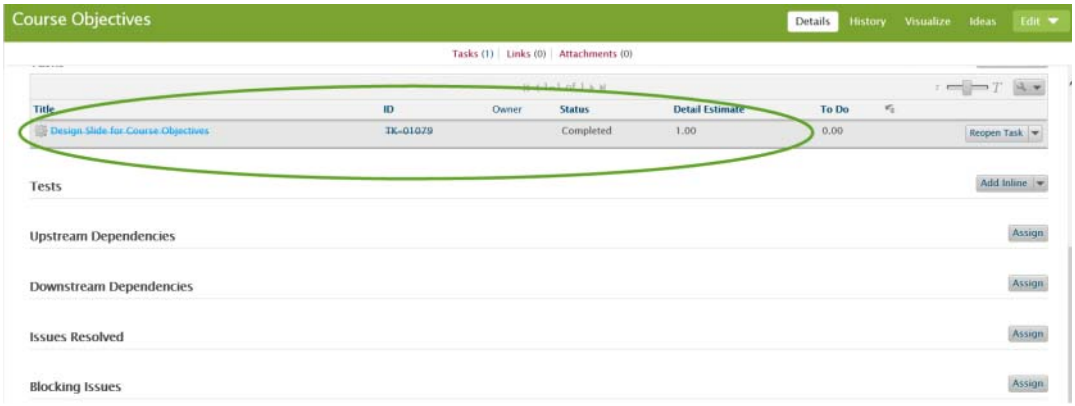


FIGURE 10.4 Adding tasks to a story in VersionOne

Manage business initiatives with epics

A good approach to product/project planning involves developing a high-level view of the product/project functionality in terms of epics before breaking it down into more detail in the form of user stories. VersionOne supports that mode of operation. Valpak, which is one of the company case studies in this book, uses this mode of planning for strategic planning for their whole business. Its senior executives identify the major initiatives that they want to accomplish in terms of epics, and their strategic planning sessions consist primarily of defining and prioritizing business initiatives as epics at this level. (They don't use the VersionOne tool, but the fundamental process is the same). The VersionOne tool provides a capability for visually planning, tracking, and managing strategic initiatives and features using the software's Epic Dashboard, Epic Bubble Chart and Portfolio Timeline. Figure 10.2 shows an example of planning high-level epics using the VersionOne tool.

Group your work items by feature groups or themes

A *theme* in agile is defined as follows:

> Generally, a theme is a categorization element for use by the product owner. A theme is made up of a set of stories grouped around some functional area, persona or some other classification criteria that is useful to the product owner for organizing requirements. It is not unusual for a theme to be aligned to a high level organizational objective that may span projects or products.[2]

[2] "Themes," http://www.agilesherpa.org/agile_coach/product_planning/theme/

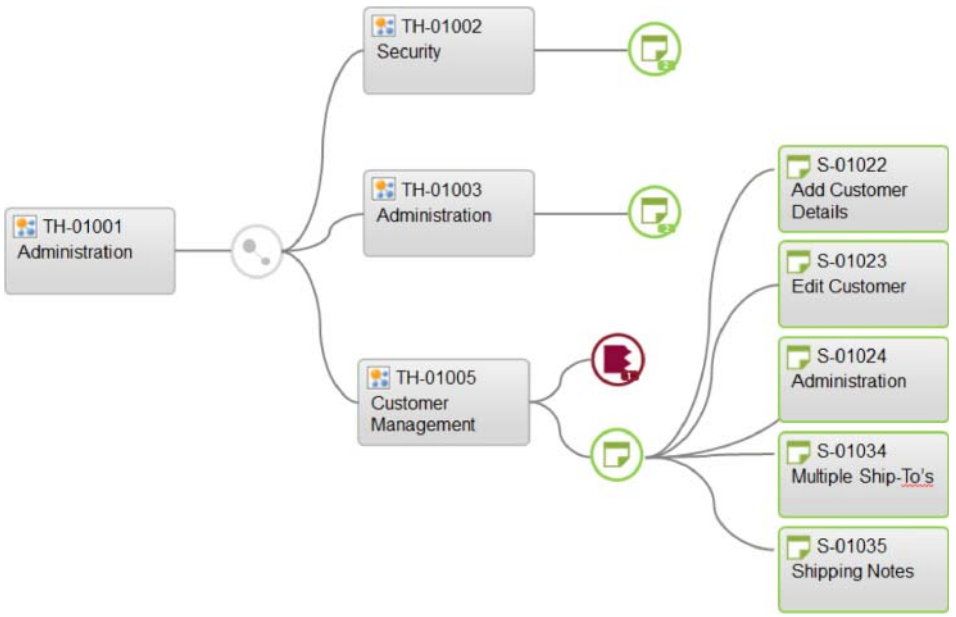


FIGURE 10.5 VersionOne theme relationship diagram

Themes are not widely used in agile projects; however, you would typically find them in larger, more complex initiatives where there is a need to roll up stories and other product backlog items by theme across multiple projects and releases for developing a higher level of visibility into how groups of features are progressing. (Note that in the VersionOne Scrum template, a *theme* is referred to as a *feature group*). Figure 10.5 shows a relationship diagram from VersionOne that can be used to visually associate themes with user stories.[3] A theme could also be a way of tying product backlog items against a high-level business initiative—for example, improve customer satisfaction could be a high-level theme that spans a number of different projects. (Note that this capability to view a relationship diagram is only available in the Ultimate edition of VersionOne—not the Enterprise edition.)

Deliver according to business goals

One of the most difficult things in an agile project is to maintain the alignment of all of the individual project requirements and tasks with high-level business goals across a number of projects. For example, there is a major financial services company in the Boston area that implemented agile on a very broad scale throughout all of the company; and, although the implementation was successful at a project and development level, many of the senior executives were unhappy that they no longer had visibility into how all the project activities were aligned with their business objectives and goals.

[3] "Agile Product Planning."

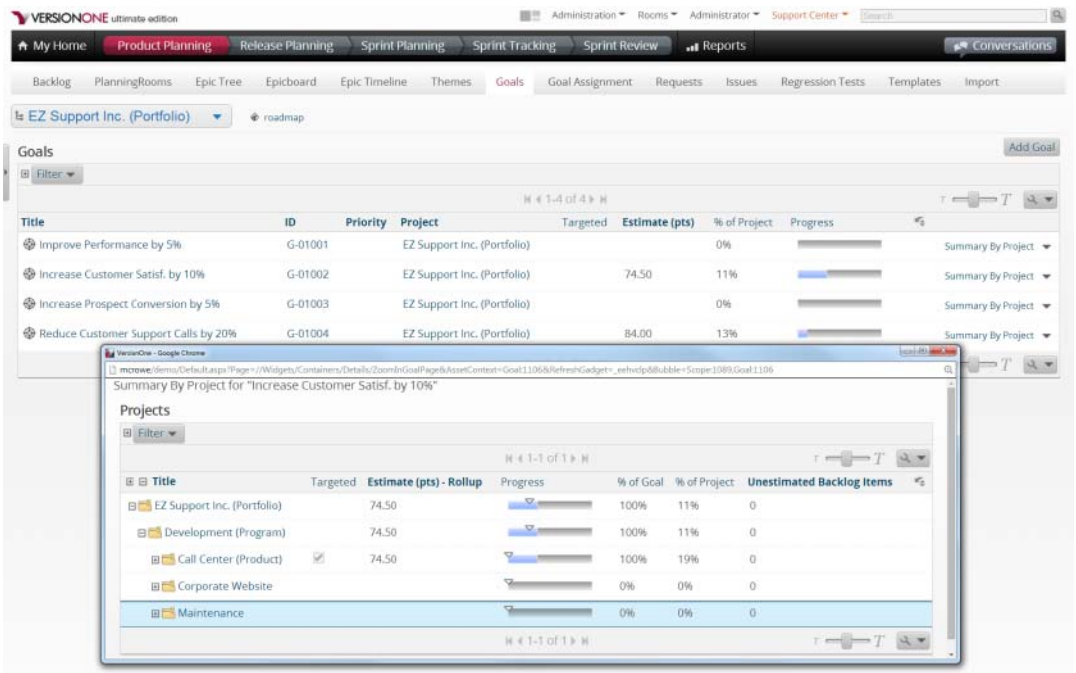


FIGURE 10.6 VersionOne business goal alignment

It's easy to see how a very fast-paced agile development process can get too far out in front of the business planning if the business planning effort isn't also streamlined and fast-paced. The VersionOne tool provides a nice capability for defining business goals and aligning those goals with product backlog items similar to themes that were previously described making it easier for agile teams to understand the full context and business importance of what they're working on. Figure 10.6 shows an example of how this would be done using the VersionOne tool.[4] Clicking any of the business goals in the Goals list would produce a pop-up window to show all of the project activities associated with that business goal.

RELEASE AND SPRINT PLANNING

Release planning in agile is typically similar to product/project planning and involves the same kind of activities but generally will be at a more tactical and less strategic level of planning. For example, you would expect that at least the epics would be defined and prioritized at the product or project level and release level planning would be concerned with further defining those epics if necessary, resolving

[4]Agile Product Planning, http://www.versionone.com/product/agile-planning-software/

any significant uncertainties, and prioritizing what should be included in a given release. There are really three levels of product backlog that need to be managed in a large agile project:

- The overall product backlog consists of all the items to be included in the project.
- The release backlog consists of all items to be included in a given release.
- The sprint backlog consists of all items to be included in a given sprint.

Of course, for small projects, release planning might not be done at all if there is no need to break the project into releases. In that instance, you might have only product/project level planning and sprint planning, with no release planning in between. Release planning, if it is done at all, is kind of a hybrid approach between sprint planning and product/project planning, as shown in Table 10.1.

What typically happens in release planning is that the product owner and the project team try to determine what can reasonably be included in the release in order to complete the release in a given amount of time. That involves some negotiation similar to sprint planning. Of course, release planning is also like product/project planning; it isn't done once and put on the shelf. It would typically be revisited as often as necessary while the release is in progress. For example, you would typically do release planning once at the beginning of the release, and you might update it at the end of each sprint if the results of that sprint caused a change in the release plan.

Release planning/sprint planning capabilities

In the VersionOne tool, release planning is probably more similar to sprint planning than it is similar to product/project planning because the product/project planning capabilities include more capabilities for linking to higher-level planning functions, such as goals, issues, and feature groups. A summary of the VersionOne capabilities to support each level of planning is shown in Table 10.2.

Sprint detail planning

The product backlog planning capability is very similar in the release planning level and the sprint planning level—both of those levels have the capability to drag-and-drop the items to be included in the release backlog or sprint backlog and to reorder the backlog based on priority. There are typically two differences between release planning and sprint planning:

- One difference is that sprint planning typically includes prioritization of defects, in addition to epics and stories. Defects that cannot be resolved in the sprint that they are discovered in would be entered into the product backlog for later resolution as they are discovered and they would be prioritized for resolution in the next sprint planning session.
- The big difference in the sprint level is the addition of the detailed task planning. A sprint planning meeting is typically divided into two parts:
  - The first part is focused on prioritizing the stories and defects to be included in the sprint, resolving any questions and issues with the stories, finalizing the stories and defects to include in the sprint, and defining the overall goal to be accomplished in the sprint.

- Once the stories to be included in the sprint have been defined, the second half of sprint planning is at a more detailed level to define the tasks associated with the stories and to assign those tasks and stories to people on the team.

The VersionOne tool supports the need for detailed planning by providing a capability to define and associate development tasks with stories and defects as well as assigning stories, tasks, and defects to developers.

TABLE 10.1 Characteristics of Levels of Planning

Planning Aspect	Sprint Planning	Release Planning	Project/Product Planning
Typical Planning Goal	Accurately define and commit to the items that can be included in the sprint (fixed time-box).	Estimate the time required to complete the release and items to be included in the release.	Define the vision and high-level goals for the project. Further define the epics and stories to be included in the project if necessary. Optionally, there may be a need to estimate the scope of the effort.
Time Schedule	The time for completing the sprint is typically fixed.	The time schedule for completing the release will generally be fixed, but in some cases, it may be only an estimate and could be subject to changes while the release is in progress.	There may or may not be an attempt to pin down a schedule estimate for completing the project.
Items to be Included	Items that are included in the sprint are fixed and are not allowed to change once the sprint is in progress.	Items to be included in the release will generally be defined at least at the epic level, but may be somewhat subject to change and further definition as the release is in progress.	The items to be included in the project may only be defined at a high level and may be somewhat open-ended.
Level of Uncertainty	Any uncertainty or issue that cannot be easily resolved during the course of the sprint must be resolved prior to starting the sprint or the items should be deferred.	Some effort must be put into resolving uncertainty and issues associated with items to be included in the release; however, the level of tolerance is much higher than the sprint level.	There may be very limited or no significant effort to resolve uncertainties and issues with items at the product or project level.

TABLE 10.2 Summary of VersionOne Planning Capabilities

Planning Capability	Sprint Planning	Release Planning	Project/Product Planning
Backlog Planning	Capability for defining what is included in the sprint backlog.	Capability for defining what is included in the release backlog.	Capability for defining what is included in the overall product backlog for the project.
Scheduling Capability	Not applicable—time for completing the sprint is typically fixed.	Release forecasting capability to project time required for release.	Similar to release level
Other Capabilities	**Detail Planning**—Capability to assign tasks to stories and tasks and stories to team members.	**Team Scheduling**—Capability to allocate release items to multiple teams.	High-Level Planning Capabilities ■ Epic Tree ■ Feature groups ■ Goals ■ Requests and issues

SPRINT TRACKING

The area of sprint tracking is where I believe agile project management tools have a huge advantage over traditional project management tools like Microsoft Project. With a traditional project management tool like Microsoft Project, there are two modes of using the tool:

1. Planning
   - Defining the structure of the project (work breakdown structures, Pert charts, Gantt charts, etc.)
   - Doing what-if analysis to explore trade-offs such as adding or subtracting resources
   - Estimating the costs and schedules for completing the project
2. Tracking progress
   - Entering progress information to update project status
   - Reestimating completion of the remaining project items based on progress to date

From my experience, most people use Microsoft Project as a planning tool and very few people use it for tracking progress because it is so difficult to use for tracking progress. The reason tracking is so difficult is that Microsoft Project is a desktop tool designed to be used primarily by a single project manager for planning and managing the project. In order to implement a tracking capability with Microsoft Project, a project manager would have to either:

1. Gather all the progress information and enter it into Microsoft Project.
2. Use Microsoft Project Server to allow people on the project to enter progress information directly.

For that reason, Microsoft Project plans typically don't go down to a low level of detail to include such things as individual developer tasks and resolution of defects.

Agile project management tools are designed around a different kind of environment.

- Instead of having a project manager who is the focal point for planning the project and tracking progress using a single desktop tool, everyone on the project team has access to a shared tool and has the ability to plan their own tasks and report progress directly in real time.
- Because the tool is shared by everyone on the team, the tool provides the ability to plan and track more detailed tasks—each individual on the team can enter his/her own tasks and update the progress of those tasks directly in the tool.

That makes progress reporting in an agile project much easier and more accurate. Information can be collected in real-time or at least on a daily basis, and the tool will automatically roll up progress reporting information across the project and across multiple teams and projects as necessary.

Kanban boards

One of the widely used capabilities for tracking progress in a sprint is a *Kanban board* (Called a *story board* in VersionOne). Figure 10.7 shows an example of a Kanban board that is used in sprint tracking.

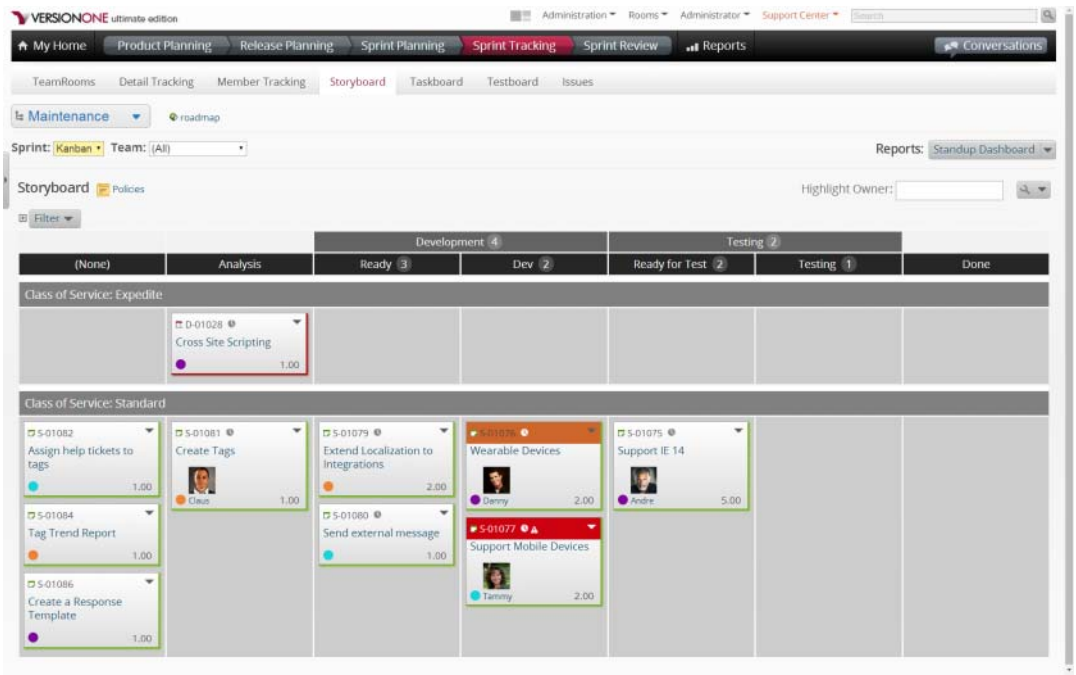


FIGURE 10.7 VersionOne Kanban board

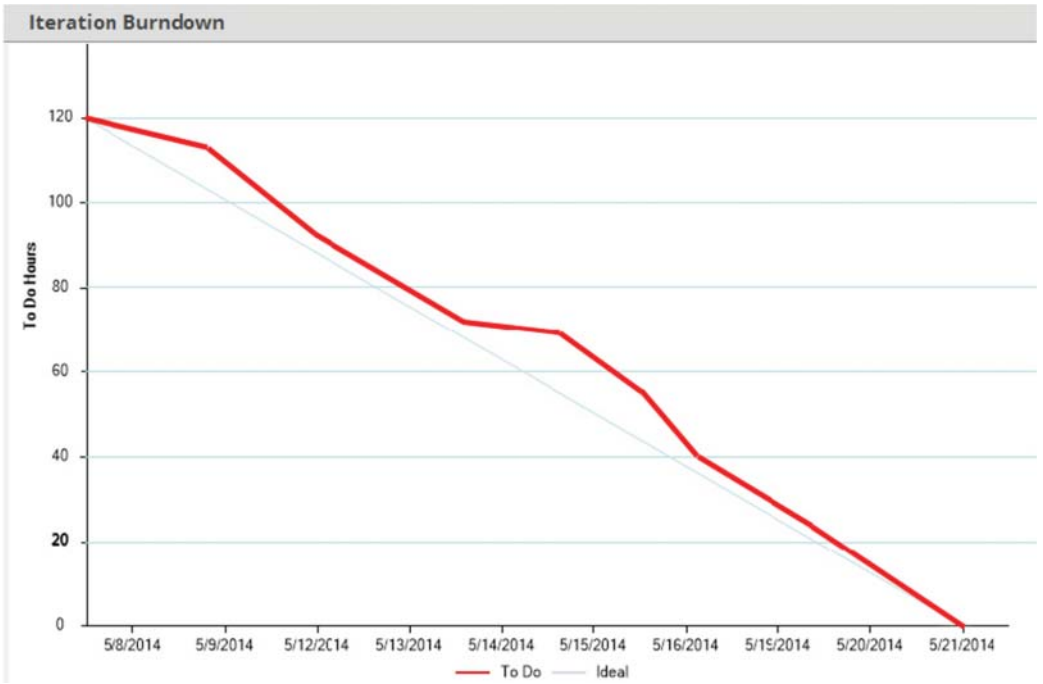


FIGURE 10.8 VersionOne burn-down chart

The Kanban board shows the progress of items in the sprint through the stages of completion that are defined for the sprint. The stages of completion can be customized and the items displayed in the Kanban board can be selected to show different views of information and to include different levels of detail.

Burn-down charts

A very widely used capability for tracking progress is a burn-down chart as shown in Figure 10.8 (also see Figure 7.3 in Chapter 3).

It is called a burn-down chart because it shows the items that have been completed in the sprint versus the items remaining to be completed. The highest point on the vertical access is the total amount of work to be completed in the sprint. That work is typically expressed in terms of story points but some teams may choose to track progress in terms of work hours as well as story points. A burn-down chart shows a graphical representation of progress in the sprint—as items are completed, they are burned down until the work remaining to be done reaches zero.

- A diagonal line shows the ideal progress if the sprint is to be completed on time.
- If progress is lagging behind schedule, the work remaining to be done will be above the diagonal line.

- If progress is going faster than scheduled, the work remaining to be done will be below than the diagonal line.

A sprint is like a mini-project; you track progress in a sprint in generally the same way as you would track a large project, but because it is so small and the items tracked are at a much lower level of detail, progress can be tracked much more accurately. Of course, accurate tracking of progress is dependent on having a clearly defined definition of "done" that the team agrees to follow. For example, the definition of "done" might include that an item has been fully tested and reviewed by the product owner.

Although the primary focus of tracking progress in an agile project is at the sprint level, agile project management tools such as VersionOne roll up progress information to higher levels automatically to allow tracking progress at different levels. For example, you could have:

- A release burn-down that shows the rate of completing items included in the release
- An overall project burn-down that shows the rate of completing all items included in the project

It's easy to see how the detailed tracking information from the sprint tracking could feed into a much higher level enterprise-level management tool. For example, Harvard Pilgrim Healthcare, which provided a case study for this project, uses Rally, which is another widely-used agile project management tool similar to VersionOne to roll-up progress information for a very large, enterprise-level project involving over 100 teams. It's hard to imagine how an effort that large and complex could be managed without a tool like this.

SUMMARY OF KEY POINTS

The VersionOne tool is a representative example of agile project management tools. It provides a very complete capability for planning and tracking a project at different levels.

- Product/Project Level
- Release Level
- Iteration/Sprint Level

It also provides capabilities for scaling projects to enterprise levels and integration with enterprise-level management functions such as product/project portfolio management.

DISCUSSION TOPICS

Complete the following VersionOne lab exercise:

1. Create an online VersionOne demo account to use the VersionOne tool.
2. Enter the product backlog given in Table 10.3 into the VersionOne tool.

3. Assign the backlog items to releases and sprints as follows:

- Release 1—Introduction, Course Objectives, and Agile Overview
- Sprint 1—Introduction and Course Objectives
- Sprint 2—Agile Overview
- Release 2—Agile Fundamentals
- Sprint 3—Agile History, Values, and Principles
- Sprint 4—Agile Benefits and Obstacles to Becoming Agile
- Release 3—Scrum Overview
- Sprint 5—Scrum Roles
- Sprint 6—Kanban Process Overview
- Sprint 7—Scrum Methodology
- Sprint 8—Time Boxing
- Sprint 9—General Scrum/Agile Principles

TABLE 10.3 Example Product Backlog

Title	Estimate
1. Introduction, Course Objectives, and Agile Overview	**8.00**
Introduction and Course Objectives	2.00
Introductions	1.00
Course Objectives	1.00
Agile Overview	6.00
What Is Agile?	3.00
Agile Perception versus Reality	3.00
2. Agile Fundamentals	**17.00**
Agile History, Values, and Principles	11.00
Agile Manifesto Values	3.00
Agile Manifesto Principles	8.00
Agile Benefits and Obstacles to Becoming Agile	6.00
Agile Benefits	3.00
Obstacles to Becoming Agile	3.00
3. Scrum Overview	**34.00**
Scrum Roles	3.00
Scrum Master Role	1.00
Product Owner Role	1.00
Team Role	1.00
Kanban Process Overview	10.00
What Is Kanban?	1.00
Differences Between Push and Pull Processes	2.00
Differences Between Kanban and Scrum	2.00
WIP Limits in Kanban	1.00
Theory of Constraints	2.00
Kanban Boards	2.00
Scrum Methodology	8.00
Time Boxing	3.00
General Scrum/Agile Principles	10.00

11 Understanding Agile at a Deeper Level

AN AGILE PROJECT MANAGER needs to understand agile at a deeper level in order to apply it to different situations effectively. The key to that is to develop a systems thinking approach to understand agile principles and practices at a deeper level. In order to develop that kind of systems thinking approach, it is valuable to understand the roots of agile and how agile thinking evolved. The roots of agile go fairly deep, but there are two major sources that had the most impact on its development:

- Total quality management (TQM) was probably the strongest factor in influencing the agile approach to quality.
- Lean manufacturing was probably the biggest factor in influencing agile process thinking.

Each of those influences will be discussed in this chapter.

SYSTEMS THINKING

BusinessDictionary.com defines *systems thinking* as follows:

> Practice of thinking that takes a holistic view of complex events or phenomenon, seemingly caused by myriad of isolated, independent, and usually unpredictable factors or forces. Systems Thinking views all events and phenomenon as 'wholes' interacting according to systems principles in a few basic patterns called systems archetypes. These patterns underlie vastly different events and phenomenon such as diminishing returns from efforts, spread of contagious diseases, and fulfillment in personal relationships.

> Systems Thinking stands in contrast to the analytic or mechanistic thinking that all phenomenon can be understood by reducing them to their ultimate elements. It recognizes that systems ('organized wholes') ranging from SOAP bubbles to galaxies, and ant colonies to nations, can be better understood only when their wholeness (identity and structural integrity) is maintained, thus permitting the study of the properties of the wholes instead of the properties of their components.[1]

In the context of agile, *systems thinking* means understanding the principles behind the methodology rather than just focusing on the mechanics of how the methodology works and understanding how those principles interact with the overall project and business environment that they are part of. Why is systems thinking important? It allows you to see things in an entirely different perspective:

- You see the whole rather than the pieces and understand their relationship. In an agile implementation you see the business as a large ecosystem and see the development process as only one component of that ecosystem and you begin to better understand how the two are interrelated to each other.
- Within an agile development process, you begin to better understand how all the components of that process work together to make the overall process more effective and instead of following the process rigidly and mechanically, you see it as a much more dynamic process where each component of the process may need to be adjusted to fit the situation.

Binary thinking is the antithesis of systems thinking. Instead of seeing the real complexity that is inherent in many situations, people who engage in binary thinking are sometimes looking for a simple, cause-effect explanation for something that isn't really very simple at all:

- They tend to see the agile values and principles in black-and-white terms, as absolute statements, rather than relative statements that need to be interpreted in the context of the situation as they were intended to be.
- They see the relationship of agile and more traditional plan-driven approaches as either-or, mutually exclusive choices (Either you're agile or you're not) and they may see these approaches as competitive with each other rather than seeing them as potentially complementary.

That sort of narrow thinking has led to many stereotypes, myths, and misconceptions about what agile is, and also about what traditional project management is. We need to rethink what agile is as well as rethink what traditional project management is to see them in a new light as potentially complementary rather than competitive approaches. Systems thinking is the key to that.

[1] "What Is Systems thinking?" http://www.businessdictionary.com/definition/systems-thinking-ST.html#ixzz2z3A07AVt.

System thinking is closely related to the idea of a learning organization. Business Dictionary.com defines a *learning organization* as follows:

> Organization that acquires knowledge and innovates fast enough to survive and thrive in a rapidly changing environment. Learning organizations (1) create a culture that encourages and supports continuous employee learning, critical thinking, and risk taking with new ideas, (2) allow mistakes, and value employee contributions, (3) learn from experience and experiment, and (4) disseminate the new knowledge throughout the organization for incorporation into day-to-day activities.[2]

Systems thinking provides a mechanism to understand the dynamics behind how an organization works at a deeper level. The culture of a *learning organization* creates an environment where that information is used for ongoing, continuous improvement. Adopting a *systems thinking* approach and becoming a learning organization are two of the most important aspects of achieving enterprise-level agility.

INFLUENCE OF TOTAL QUALITY MANAGEMENT (TQM)

In order to understand agile at a deeper level, it is useful to understand the roots of agile and how those principles have evolved in different environments. Many of the agile principles related to quality have their roots in the philosophy of total quality management (TQM). The TQM philosophy originated from the ideas of W. Edwards Deming and others. Dr. Deming was an American statistician who was credited with the rise of Japan as a manufacturing nation. His principles transformed the Japanese automotive industry into developing very-high-quality products that gained significant market share against American automotive manufacturers in the 1970s and 1980s (see Figure 11.1).

Dr. Deming's original 14 points can be summarized into five major areas that have a significant impact on an agile project management approach. The following are a summary of some of Dr. Deming's original 14 points that the TQM philosophy is based on that form the roots of today's agile approach for software development. According to Deming, "The 14 points all have one aim, to make it possible for people to work with joy."[3]

1. Cease dependence on inspection
2. Emphasis on the human aspect of quality
3. The need for cross-functional collaboration
4. Importance of leadership
5. Ongoing continuous improvement

[2] "What Is a Learning Organization?" http://www.businessdictionary.com/definition/learning-organization.html#ixzz2z3C2Bv5s.
[3] Alexander Laufer, *Mastering the Leadership Role in project management: Practices that Deliver Remarkable Results* (Upper Saddle River, NJ: Pearson Education, 2012).

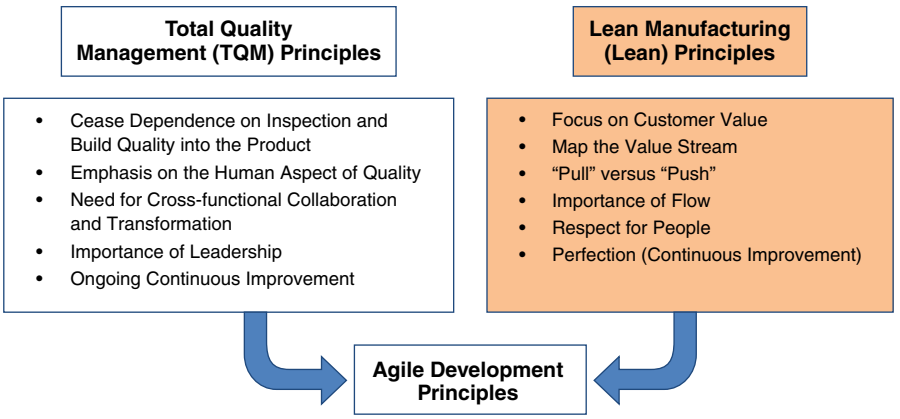


FIGURE 11.1 The roots of agile practices

Each of these points and how it impacts an agile project management approach will be discussed in the following sections.

Cease dependence on inspection

Deming's Principle:	Point #3—"Cease dependence on inspection by building quality into the product in the first place."
Manufacturing Problems Prior to TQM	**Manufacturing TQM Approach**
Prior to TQM, manufacturers relied heavily on quality control inspectors to detect problems at the end of the assembly line. The problems with that approach are: ■ If problems are not detected until the end of the assembly line, it can be expensive to go back and rework or scrap the product at that point. It also can have a significant impact on cycle time to wait for a product to be reworked before it can be shipped.	A better approach is to go upstream in the process, find the sources of error that contribute to the defects, and eliminate those sources of error at the source. That approach results in: ■ Better quality because the quality is designed into the product from the beginning, and defects are prevented before they happen

(*Continued*)

Manufacturing Problems Prior to TQM	Manufacturing TQM Approach
■ Any inspection approach like this relies heavily on sampling; and, with a limited number of inspectors, it's very time-consuming to do a very large sample size. For that reason, it's difficult to fully test every possible product or situation that could result in a defect. As a result, some defects are bound to slip through and result in very poor quality, as seen by the customer. ■ Quality is seen as the responsibility of the inspectors, who are perceived as the *enforcers*. As a result, the people who are producing the products may not feel fully responsibility for the quality of the products they produce. ■ It is expensive to employ enough quality control inspectors to perform this function completely, and even with lots of inspectors, it still might not be very effective.	■ Reduced costs because it relies less on inspection to find problems ■ Greater pride of ownership by the workers who are primarily responsible for producing the product

Implications for Agile Development:

An agile development approach recognizes and incorporates this principle. Instead of performing QA testing sequentially with development and relying heavily on a typical quality assurance approach to detect problems after development has been completed and sending the product back for rework to fix defects (bugs), it is far more effective to make quality an integral part of the development process, do it more concurrently with development, and prevent the bugs (defects) from happening at the source. This approach also results in reduced costs because it relies less on inspection (QA) to find problems after the product development is complete.

Agile makes producing quality products the responsibility of the team developing the product, which typically includes QA testers. It's not someone else's responsibility (like a separate QA department) to ensure that the product is free of defects. This increases pride of ownership of everyone on the team and results in much higher quality products.

Emphasis on the human aspect of quality

Deming's Principles:

- Point #6: "Institute training on the job."
- Point #8: "Drive out fear."
- Point #12: "Eliminate barriers to pride of workmanship."
- Point #13: "Institute education and self-improvement."

Manufacturing Problems Prior to TQM

In the early days of manufacturing, *sweatshops,* where people worked under oppressive conditions, were quite common. It's obvious that people who are overworked and who aren't respected and recognized for the importance of the work they do are probably not going to be highly motivated and are likely to produce a much lower quality product. In particular,

- People who are fatigued from working very long hours in a less than ideal working environment are prone to make errors.
- People with narrowly defined and repetitive jobs, who only have a limited responsibility for a small portion of the overall product (such as putting the bolts on a wheel on an automobile), may have difficulty feeling ownership for and taking pride in the overall product they are producing. That may affect the quality of the product they produce.
- People who are primarily motivated by fear of the consequences of *not* performing a task well are generally not going to work as effectively as people who are *positively* motivated to produce a high quality product because they take pride in their work.
- No one wants to be a *cog in a wheel*. People want to see a higher purpose and value in the work they do, and they want to be respected and recognized for their work.

Manufacturing TQM Approach

Manufacturing companies learned a long time ago to pay attention to the human aspects of producing quality products:

- Improving working conditions and automating menial, repetitive tasks as much as possible, rather than relying on people to perform those tasks, uses the skills of people more effectively and removes a major source of errors and defects.
- Engaging people at all levels of production through *quality circles* and other mechanisms so that they feel responsibility for the quality of the overall products they produce, rather than just their own particular role in producing the product, will ultimately lead to more pride of ownership and higher quality products.
- Providing training and education to all employees, building their skills to perform the process at a higher level, and empowering them to recognize and suggest opportunities for improvement in the process are essential for ongoing process improvement.

(*Continued*)

Implications for Agile Development:

An agile software development approach recognizes this by:

- Making respect for people a very important value.
- Fully engaging everyone on an agile development team as an equal contributor to the success of the product.
- Putting a high level of emphasis on the training and skill of individuals to exercise good judgment in how the process is executed rather than relying on highly prescriptive, predefined processes to tell people what to do.
- Substituting positive motivation and leadership for traditional command-and-control management. The Scrum Master on a team is a facilitator, not a directive manager, and should empower everyone on the team to be fully engaged in the process.
- Eliminating sweatshops and Death March projects where people are forced to work excessive amounts of overtime to meet arbitrary schedule commitments and instead working at a *sustainable pace*.

The need for cross-functional collaboration and transformation

Deming's Principles:

- Point #2: "Adopt the new philosophy."
- Point #9: "Break down barriers between departments."
- Point # 14: "The transformation is everyone's job."

Manufacturing Problems Prior to TQM	Manufacturing TQM Approach
Prior to TQM, responsibilities were typically split across different organizations (engineering, production, quality, etc.). As a result: ■ Each organization was typically focused on their individual objectives, and the responsibility for the overall effectiveness and quality of the process was fragmented.	Moving to a TQM approach required a major shift in thinking using *systems thinking* to see the whole business from a much broader process perspective (rather than a hierarchical organizational perspective) and to develop a well-integrated cross-functional approach across all of the organization. Although this can

(*continued*)

(*Continued*)

Manufacturing Problems Prior to TQM	Manufacturing TQM Approach
■ It can be very difficult to break down these barriers to develop a much more integrated and unified approach.	be difficult to achieve, the results are significant: ■ It eliminates conflicting goals among organizations and develops an integrated cross-functional approach that leads to much higher levels of productivity and efficiency. ■ It enables everyone in the company to see an overall vision of how the products and projects they're producing provide value to leverage the company's business success and how their individual role contributes to that goal.

Implications for Agile Development:

An agile development approach addresses this by emphasizing self-sufficient and autonomous cross-functional teams as the focal point for responsibility and decision making to break down organizational barriers at the project level. Even that doesn't go far enough, in many cases.

A major problem with the implementation of an agile development process is that it is often perceived as just a software development process owned by the development organization. People incorrectly ignore the need for organizational transformation and senior management commitment to make it an integral part of the way the business operates to make it fully successful.

Importance of leadership

Deming's Principle: Point #7: "Institute leadership"—Supervision should help people and machines do a better job. Supervision of management is in need of overhaul as well as supervision of production workers.

Manufacturing Problems Prior to TQM	Manufacturing TQM Approach
Traditional command-and-control styles of management may not be very effective, even in a manufacturing environment. There are a couple of significant problems in that approach: ■ It doesn't fully empower and engage employees to take an active role in processes that are more self-managed. ■ It can be very demotivating to employees to work in that kind of environment where their skills are not recognized and valued and without more effective leadership.	Truly inspirational leaders help people see the higher-level purpose and vision for their work. As a result, people ■ Are much more highly motivated; ■ Feel more ownership of the products they produce; ■ Work more effectively; and ■ Use their initiative with less need for direct supervision.

Implications for Agile Development:

In a software development environment where creativity and innovation are extremely important, traditional command-and-control management styles may stifle that initiative and creativity.

Ongoing continuous improvement

Deming's Principles:
- Point #1: "Create constancy of purpose toward improvement of products and services with the aim of becoming competitive, staying in business, and providing jobs."
- Point #5: "Improve constantly and forever. Constantly improve quality and productivity in order to constantly decrease costs."

Manufacturing Problems Prior to TQM	Manufacturing TQM Approach
Processes that rely heavily on a reactive approach based on correction of defects for quality control, rather than a more proactive approach based on	With the advent of TQM and Six Sigma in the 1980s and early 1990s, companies learned that they could "push the envelope" much further into developing higher levels

(*continued*)

(*Continued*)

Manufacturing Problems Prior to TQM	Manufacturing TQM Approach
prevention of defects and continuous improvement, will not significantly improve their defect rate because they have not removed the root cause of the defects.	of quality that were never believed possible before that time. It required a totally different and much more systemic approach to eliminate the source of defects and prevent them from happening in the first place, rather than fixing them after they've happened.

Implications for Agile Development:

Continuous improvement is a major focus of all agile methodologies. The need for continuous improvement is done through retrospectives at the end of each sprint or iteration. Because the iterations are relatively short, learning and process improvement can take place rapidly.

INFLUENCE OF LEAN MANUFACTURING

Where TQM provides a foundation of how to integrate quality into the design of products, lean manufacturing principles (see Figure 11.2) complement and go beyond that by developing a stronger focus on *maximizing customer value and providing guidance on how to improve and streamline processes to eliminate wasteful inefficiencies*.

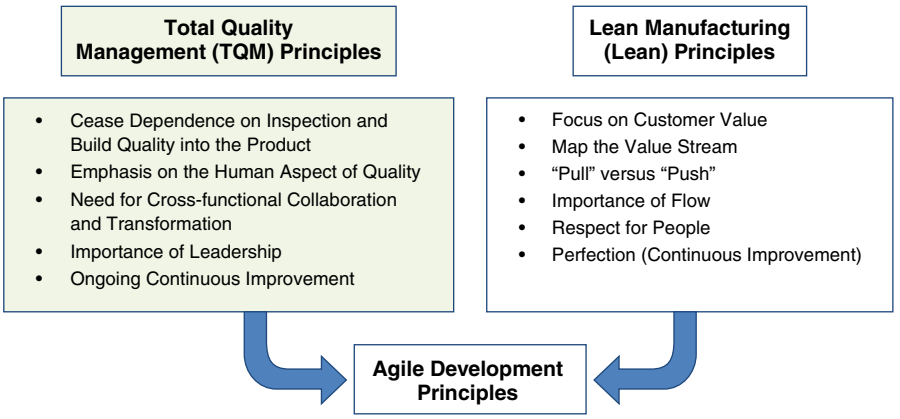


FIGURE 11.2 The roots of agile principles (lean)

Lean manufacturing or lean production, which is often simply known as Lean, is defined as:

> A systematic approach to identifying and eliminating waste through continuous improvement by flowing the product at the demand of the customer.[4]

Lean considers the expenditure of resources for any goal other than the creation of value for the end customer to be wasteful and a target for elimination. Value is defined as any action or process that a customer would be willing to pay for. Agile is based on taking that same thinking from lean manufacturing and applying it to a software development process. It involves looking at a software development process and making critical decisions about whether each activity in the process adds value. There are three kinds of work in any process:

1. *Value-added:* Process steps that produce value the customer is willing to pay for or are essential to directly meeting customer requirements
2. *Non-value-added:* Process steps that are not directly required to produce customer value but are required for other reasons, such as meeting regulatory requirements, company mandates, and legal requirements,
3. *Waste*: Process steps that consume resources but produce no value in the eyes of the customer

Applying these concepts to a software development lifecycle model requires evaluating the various steps in the process and making a judgment of whether they really produce value in the eyes of the customer or not. Dr. David Rico provides a definition of *lean systems engineering* as follows:

> Lean (lēn): Thin, slim, slender, narrow, adequate, or just-enough; without waste
>
> - A **customer-driven** systems engineering process that delivers the maximum amount of **business value**
> - An economical systems engineering way of **planning** and **managing** the development of complex systems
> - A systems engineering process that is **free of excess waste**, capacity, and non-value-adding activities
> - **Just-enough**, just-in-time, and right-sized systems engineering **processes**, **documentation**, and **tools**
> - A systems engineering approach that is **adaptable to change** in customer needs and market conditions[5]

[4]Lean Manufacturing Guide, http://www.leanmanufacturingguide.com/.
[5]David F. Rico, "Lean and Agile Systems Engineering," http://davidfrico.com.

INCOSE[6] has developed a list of systems engineering enablers to support the six key principles of lean:

Lean Principle	Enablers
Value	Focus on delivering customer value: ■ Use a defined process for capturing requirements focused on customer value. ■ Establish the value of the end product or system to the customer (What are the business objectives?). ■ Frequently involve the customer.
Map the Value Stream	Use a well-defined methodology for executing projects: ■ Poor planning is the most notorious reason for wasteful projects. ■ Plan to develop only what needs developing. ■ Plan leading indicators and metrics to manage the project.
Pull	Tailor the process to the risks and complexity of the project to achieve maximum efficiency: ■ The *pull principle* promotes the culture of tailoring tasks and pulling them and their outputs based only on legitimate need and rejecting others as waste. ■ Pull tasks and outputs based on need, and reject others as waste.
Flow	Eliminate bottlenecks that are likely to obstruct or delay progress: ■ In complex programs, opportunities for the progress to stop are overwhelming, and it takes careful preparation, planning, and coordination effort to overcome them. ■ Clarify, derive, prioritize requirements early and often during execution. ■ Frontload the architectural design and implementation. ■ Make progress visible to all. ■ Use the most effective communications and coordination practices and effective tools.
Respect for People	Build an organization based on respect for people: ■ Nurture a learning environment. ■ Treat people as most valued assets.

[6] "International Council on Systems Engineering: Lean Systems Engineering Working Group," http://cse.lmu.edu/about/graduateeducation/systemsengineering/INCOSE.htm.

(*Continued*)

Lean Principle	Enablers
Perfection	Strive for excellence and continuous improvement in the software development processes: ■ Use lessons learned from past projects for future projects. ■ Develop perfect communication, coordination, and collaboration policy across people and processes. ■ Use effective leadership to lead the development effort from start to finish. ■ Drive out waste through design standardization, process standardization, and skill-set standardization. ■ Use continuous improvement methods to draw best energy and creativity from project teams.

Each of these topics is discussed in more detail in the following sections:

Customer value

Many businesses focus heavily on financial results rather than customer value as a primary goal. Without understanding the cause-and-effect relationships that drive those financial results, you're always in reactive mode. When the financial results for the current quarter go down, there's a lot of scrambling around to figure out what went wrong and what caused that to happen and then trying to fix the problem. That's a very reactive approach.

A more proactive approach is to develop an understanding of the factors that drive financial results for your business and focus on those factors. If you focus on customer value and the factors that influence customer value and you do that successfully, the financial results should follow; it's a much more proactive, reliable, and consistent approach for managing a business successfully. The idea is that managing the inputs to a process and managing the process itself is always a better approach than simply trying to manage the outputs of a process.

As an example, I once worked for a company that was known for putting enormous pressure on project teams to meet schedule deadlines. If a project was unsuccessful, there was a good chance that the whole project team might be fired. The problems with that approach should be very apparent—taking a *brute force* approach to try to force the outputs of a process to produce the desired results without an understanding of how the process works or the factors that influence the outputs is not likely to be very effective.

Map the value stream

An important principle of both lean manufacturing and lean software development is to map the value stream. This process basically involves starting from the point that the product or service is delivered

to the customer (either internal customer or external customer) and working backward from that point to map all the process steps that led up to fulfilling that customer value. The next step is to identify and differentiate steps that produce value to the customer from steps in the process that produce no value to the customer and may constitute waste.

An important factor is to identify *waste*. Mary Poppendiek has translated the seven wastes found in a manufacturing process to the equivalent seven wastes found in software development:[7]

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

Naturally, in order to manage the sources of waste in a process, you have to have a somewhat defined and documented process that is followed fairly consistently.

Pull

One of the key differences associated with lean is the difference between a *push approach* and a *pull approach*—this difference is also found in most agile approaches. In a traditional manufacturing process approach, production output is forecast, inventory is stocked at various points, and raw materials are then pushed through the process to fulfill that forecast. This type of process has been the predominant approach used in manufacturing for many years to maximize the efficiency of the production equipment used in the process. This approach has three potential deficiencies:

1. The forecasting process requires an intelligent guess at what the customer demand will be well in advance of when it is actually expected to be delivered from production. This is very difficult to do accurately and is fraught with lots of potential problems.
2. The process is very difficult to adjust—if there is a change in customer demand, it can take a considerable amount of time and effort to re-plan the entire process to adjust to that change. There also may be a considerable lag associated with restocking material to support the revised plan.

[7]Tom and Mary Poppendiek, *Lean Software Development—An Agile Toolkit* (Reading, MA: Addison-Wesley, 2003), p. 4.

3. If the forecast is wrong, there are significant potential risks, including:
   - *Winding up with a significant amount of unusable inventory that might have to be scrapped* (In a software project, unusable inventory translates to extra features that no one needs or is likely to use that complicate the product and cause unnecessary maintenance if they are not removed.)
   - *Not having sufficient inventory to fill customer demand if the forecast is wrong* (In a software project, this translates to not having the right features to satisfy customer needs.)
   - *Having to stockpile or store inventory beyond the originally planned duration* (In a software project, this translates into undesirable product management overhead.) "I've often seen this administrative burden lumped onto the project manager who must now wade through bloated scope matrices, backlogs of change requests and unwieldy specifications."[8]

Most traditional plan-driven product development processes such as the waterfall process are based on a similar *push process*. In a software development process, requirements are equivalent to raw materials in a manufacturing process. All the requirements are gathered up front and are pushed through the rest of the development process in a sequential fashion just like a manufacturing assembly line. In a traditional plan-driven product development process, the *push approach* may have the advantage of optimizing the utilization of the resources in a development process if the requirements are relatively certain and known in advance, but that is often not the case. If there is uncertainty in the requirements, it can result in serious potential problems and inefficiencies. Those potential problems and inefficiencies are similar to those associated with a push manufacturing process:

- The product requirements for a new product development effort are essentially a *forecast* of the requirements for a product or application that a customer will need in the future. These requirements may be very uncertain and not very well-defined. Attempting to forecast (or guess at) product requirements well in advance of when the product will actually be deployed and used has even more risk than forecasting production output in a manufacturing process. In addition to the normal risks of forecasting customer demand, the customer may not really know what he/she wants without seeing the product and seeing firsthand how it works. For that reason, the requirements might easily change over that time.
- The process also can be difficult to adjust when changes in customer requirements occur. Typically, many assumptions are made about what the customer requirements are, and elaborate plans, resource assignments, and documentation will be created to support those assumptions.
- There is also a significant risk that the assumptions in the requirements are wrong and don't reflect the real needs of the customer. This may not be discovered until the final product is ready for final acceptance testing.

[8]Erik Gottesman, Personal e-mail comments on book review.

Inaccurate or changing customer requirements may result in a significant amount of lost time and effort to replan around a different set of requirements and assumptions. It might also require substantial rework. If a typical change control system is used, the process for doing that may be very cumbersome and difficult. Both lean and agile approaches avoid these problems by:

- Deferring the resolution of uncertain requirements until a decision is required (when that particular requirement is *pulled* into development for further processing). Avoiding guessing at the requirements up front and waiting until more information is known will typically result in better decisions and avoid many of the problems associated with inaccurate and changing requirements.
- Lean and agile systems openly acknowledge and are built around the assumption that requirements are uncertain and are likely to change as the project progresses. Many traditional processes do not recognize or acknowledge an appropriate level of uncertainty that is actually inherent in the requirements and attempt to superimpose a rigid control model on top of a very uncertain environment.

A *pull* system works by producing only the required amount to meet demand at each stage. In a manufacturing system, this would be characterized by a just-in-time production scheduling system. Many of the ideas for lean manufacturing came from the Toyota Production System and Kanban. If the word *Kanban* were translated literally, *Kan* means visual, and *ban* means card or board. The idea is based on inventory demand cards that are sometimes used in a manufacturing system:

> Picture yourself on a Toyota production line. You put doors on Priuses. You have a stack of 10 or so doors. As you keep bolting them on, your stack of doors gets shorter. When you get down to 5 doors, sitting on top of the 5th door in the stack is a card—a Kanban card—that says build 10 doors. Well it may not say exactly that—but it is a request to build exactly 10 more Prius doors.
>
> You pick the Kanban card up, and run it over to the guy who builds doors. He's been waiting for you. He's been doing other things to keep busy while waiting. The important thing here is that he's NOT been building Prius doors. He takes your Kanban card and begins to build doors.
>
> You go back to your workstation, and just a bit before your stack of doors is gone, the door guy comes back with a stack of 10 doors. You know that Kanban card is slid in between doors 5 & 6. You got the doors just in time.[9]

The process flow works in a similar way for the rest of the plant—when the guy who makes the doors for the Prius runs out of parts that he needs to build the doors, he has a similar Kanban card to

[9]Kanban Development Oversimplified, http://www.agileproductdesign.com/blog/2009/kanban_over_simplified.html.

request more parts from the process that provides those parts to him. The whole process flow is pulled by *actual* customer demand, rather than being pushed by a forecast of someone guessing at what they *think* the customer demand is. Of course, in many situations, this process is computerized, and there are no physical cards used to signal demand.

In an agile software development process, there is a direct analogy between Kanban cards and user story cards. User stories are high-level descriptions of capabilities that the system needs to provide. The following is an example of a user story:

> "As a banking customer, I need to be able to withdraw funds from my account through an ATM machine."

User stories are typically defined early in the project to identify the capabilities that the system must provide, with a sufficient level of detail to do only a rough estimate of the level of effort associated with each. The details of how the user story will be implemented will generally be deferred until it is time to do the design:

- Instead of attempting to define all of the requirements in detail, typically only the high-level requirements are defined upfront to the level of user stories without a lot of detail.
- Instead of treating all requirements equally, the requirements are prioritized based on their value to the customer. After they are prioritized and broken up into releases and/or iterations, the most important requirements, which are at the top of the list and ready for development, get developed first.
- Once the developer has picked up a user story to begin working on, he/she will then work directly with the user to *pull* more detail as needed to fill that requirement.

A story card is equivalent to a Kanban card in a manufacturing system and describes one particular feature that a user needs. As in the manufacturing system, physical cards may or may not be used—there are computerized tools that will automate this task and eliminate the use of physical story cards if desired. However, in many cases, physical cards may actually be used and put on a board; each developer picks up a card to start working on it, similar to the way a Kanban card works in a factory.

A Kanban development process is sometimes used as an alternative to Scrum in situations that need to be more reactive, such as managing a queue of customer service requests. In a Scrum process, there is some level of upfront planning to at least identify the items in the product backlog at a high level before the project starts, and additions to the backlog items are not allowed once an iteration has started. From that perspective, a Scrum process is only *partially* based on *pull*. At a high level, requirements are *pulled* through the process based on priority; however, within an individual iteration, once the iteration has started, requirements are *pushed* through the process.

A Kanban process is designed to be much more reactive and responsive to customer demand. The following table shows a summary of the differences between a Kanban development process and Scrum:[10]

	Scrum	Kanban
Primary Flow	Time-boxed iterations (sprints)	Continuous flow
Primary Metric	Velocity	Lead time
Work-in-Process (WIP)	WIP limited indirectly (per sprint)	WIP limited directly (per workflow state)
Addition of New Items	No new items added to the current sprint	Add new item as capacity is available
Roles	three prescribed roles (product owner, Scrum Master, team)	No prescribed roles
Status Board	Scrum board reset between sprints	Kanban board is persistent

The key difference is with regard to how the two processes manage flow.

- A Kanban process is *totally* pull and allows demand for new items to take place at any time, *including during the middle of a sprint* if capacity is available.
- A Scrum process is *mostly* pull. The product backlog is considered to be dynamic and can be adjusted as needed to meet customer needs within the constraints that (a) the capacity to handle the demand is fixed and (b) additions are not allowed once an iteration has started.

Flow

Flow is one of the most important lean principles to understand to maximize the efficiency of any process. There are a number of factors that contribute to maximizing flow, which include:

- Small batch sizes
- Just-in-time production
- Concurrent processing

Each of these topics is discussed in the following sections.

Small Batch Sizes

In a manufacturing process, it is well known that small batch sizes are much better for optimizing the flow of a process than large batch sizes. If large batch sizes are used, bottlenecks can easily develop

[10]Stephanie Stewart, Valpak Agile Overview PowerPoint Presentation

at various points in the process, and material winds up waiting at those bottleneck points, creating waste. There are at least a couple of types of *waste* associated with large batch sizes:

- Excess material inventory is used in the process, creating unnecessary inventory cost, space for storage, and additional handling costs. Mary Poppendiek uses an example of the construction of the Empire State Building in New York. The Empire State Building was the tallest building in the world and was built in a total of 20 months, including demolition of existing buildings and planning and design of the new building.[11]

  One of the most serious constraints was that there was only a limited amount of vacant real estate in the area to store the materials needed for the building. This meant that the flow of the project had to be very carefully planned. The building was built in iterations of a few floors at a time, and the arrival of material had to be scheduled meticulously to have just the right materials available at the right time to maximize the flow.

- If the inventory is perishable, it can go stale and become unusable. By *perishable,* I'm not necessarily referring to fruits and vegetables. Dell Computer is a good example. Dell builds systems for customers out of a variety of different components (disk drives, graphic cards, etc.), and those components become obsolete quickly and are constantly being replaced by newer versions. Using small batch sizes and building systems on demand as customers need them reduces the risk of winding up with too much obsolete inventory of components in the pipeline.[12]

The other major advantages of small batch sizes are[13]:

- It reduces the end-to-end cycle time (Little's Law of Queuing[14]).
- It makes waste a very hard problem to ignore as any waste in a small batch size system will cause much larger problems than when you've got inventory at hand to smooth it over. You then have to confront and fix the waste. The visual metaphor often employed is that a stream running low uncovers the rocks on its bed.

Attempting to define all the requirements for a product or application up front in a traditional development process is analogous to attempting to process large batch sizes in a manufacturing process. It's impossible to work on all the requirements at once, so bottlenecks develop at various points in the process and requirements wind up waiting to be processed. Having an excess of requirements sitting around waiting to be processed is similar to having excess inventory in a manufacturing process:

- There are handling costs associated with managing those requirements waiting to be processed—they have to be well documented and tracked or they may be forgotten and left out of the design. There is also a certain amount of overhead associated with managing changes to these requirements.

[11] Poppendiek, p. 102.
[12] Ibid, p. 12.
[13] Martin Burns, personal e-mail comments, on book review.
[14] "Principle: Little's Law," http://www.factoryphysics.com/Principle/LittlesLaw.htm.

- The requirements are also perishable—if they wait for a long time to be processed, they could easily become obsolete. If that happens, either someone winds up designing and building a product or application based on obsolete requirements, or unnecessary labor is consumed in redefining and rewriting the requirements.

An iterative development process is analogous to small production batch sizes in a manufacturing operation. By breaking up the requirements into the smallest possible units (user stories), the overall development process is likely to flow more smoothly and avoid the bottlenecks associated with traditional development processes. Of course, in actual practice, there are limits to how far it is practical to break down the requirements to optimize flow. For example:

1. *Requirements management considerations*. From a requirements management perspective, it may be necessary to group requirements into feature sets that are interrelated with each other. Those feature sets might be bigger than the effort that can be realized in a single iteration. That requires some compromises between:
   - An idealized approach where individual sets of requirements are completely processed immediately in each iteration; and
   - A more realistic hybrid approach where some of the requirements might be *pipelined* for processing and spread across more than one iteration.

     The idea of buffering some of the requirements that cannot be fulfilled immediately is called *story pipelining*.

> Story pipelining is often seen by purist Agile practitioners as strictly un-Agile as it violates the oft-held view that working software is the only thing that represents value and anything less is a cop-out. In practice, however, pipelining is often the best way to balance agility with the realities of real-world delivery constraints. You gain a measure of project progress that's still closer to real doneness in the eyes of the customer, you retain a lifecycle that encourages regular and frequent feedback, but you also recognize that complex software systems have a gestation period.
>
> Pipelining particularly recognizes requirements that are just really difficult to implement: thorny user interactions, challenging external systems dependencies, etc.[15]

2. *Testing/release and configuration management considerations*. Testing and release management considerations might require compromises from the ideal flow model:
   - Testing might want a functionality set to test against that's a relatively complete subset of functionality and will be stable for the duration of the test cycle.
   - Release and configuration management might have similar needs.

[15]Gottesman, Erik Comments on book review

Both of these issues can be overcome, but may require some rethinking of how the process works and very strong coordination of test planning with the rest of the development effort.

Just-in-Time Production

Having large quantities of raw materials waiting to be processed is inefficient because they become bottlenecks. A much more efficient process is based on just-in-time processing, when the raw materials arrive just at the right time that they are needed in production. A large business requirements document in a software development environment is equivalent to a large pile of raw materials in a manufacturing environment. Instead of developing large requirements documents that wait to be processed, it is generally more efficient to develop requirements just-in-time as needed in the software development process. For example:

- Early in the process, high-level requirements should be sufficient to do whatever level of planning is needed at that point.
- The elaboration of requirement details can be deferred until later in the process when those particular requirements are ready to enter development.

Concurrent Processing

Concurrent processing is another well-known way to improve flow in any process. In a manufacturing process, bottlenecks are much more likely to develop if there is only one path through the system and everything is sequential than if there are parallel paths available and some work can be done concurrently on different paths.

In a product development process, there are typically greater opportunities for concurrent engineering to improve the flow through the process. Here are a few examples:

- Requirements development can be overlapped with design instead of being sequential, and quality testing can also overlap with design instead of following it sequentially. This requires a much more collaborative, cross-functional approach to development, which can be difficult to achieve, but the potential payoff can be significant.
- Design teams can work on multiple iterations concurrently. This requires breaking up the design effort into iterations and requires some coordination among design teams:

> Concurrent engineering is especially useful when dealing with 'unprecedented' requirements . . . This can provide cover in situations where you have multiple integration approaches to choose from and don't know which is best (e.g., which will deliver the right performance, availability, or conform to other non-functional requirements). Similarly,

challenging UI problems are sometimes best tackled using concurrent engineering (e.g., prototyping multiple solutions and testing them with real, representative users) in a 'survival of the fittest' approach.[16]

Respect for people

In the early days of automotive manufacturing, processes were designed so that the people performing those processes did not require high levels of skill. An individual working on an assembly line could be assigned a small, repetitive task such as installing a tire on a car, which required only a minimum amount of skill and training. The primary requirement for higher levels of skill and training could be limited to a relatively few people who were responsible for designing and managing the overall process and training the workers to perform each task. There are several problems with that approach:

- No one really takes responsibility for the overall quality of the complete vehicle.
- This approach might rely heavily on quality control inspectors at the end of the line to try to find defects and send the vehicle back for rework if necessary.
- It can be a dehumanizing experience for anyone to perform that kind of limited, repetitive task.
- This approach doesn't take advantage of the complete range of skills and judgment of the people performing the tasks.

Manufacturing processes have long recognized the need to respect and empower the people performing the processes as much as possible. In a manufacturing process, having people take pride in workmanship is extremely important for achieving high levels of quality and productivity. The need for fully utilizing the capabilities of people and motivating them is even more significant in a software development process, where the overall effectiveness of the process is so critically dependent on the performance of the people. Both lean and agile methodologies seek to eliminate those problems by empowering individuals and the team as a whole to take responsibility for the overall quality of their work.

Many traditional development processes have been modeled on early manufacturing processes, where a process defines in detail the work to be done and how it should be done, and the process requires a lower level of skill to perform relatively well-defined tasks. Agile methodologies are generally much less well defined and rely heavily on the skill and training of the people performing the process to use appropriate levels of judgment and tailor them to a particular project, task, and business environment. That is a key reason why respect for people is so important in an agile environment.

[16] Ibid.

Perfection

The principle of perfection in lean manufacturing is similar to the TQM principles associated with ceasing reliance on inspection and ongoing continuous process improvement to remove defects. Defects in lean are seen as a major source of waste:

- It takes a lot of resources to inspect for defects, which wouldn't be necessary if the defects were eliminated at their sources.
- Rework and scrap can result from defective products if those defects aren't discovered until the product is at the end of the assembly line (or at the end of the development process in a software development environment).

Lean manufacturing also emphasizes continuous improvement to eliminate the waste caused by defects, just as TQM emphasizes it for improving product quality.

There is also a direct relationship with the principle of respect for people. In many cases, the people performing the process are the first ones to recognize opportunities for improvements in the process to prevent defects and/or to make the process more efficient. Unfortunately, many times they are not empowered to suggest or make those changes. Lean and agile methodologies recognize that and therefore are not rigidly defined or prescriptive—they provide some fundamental principles and practices that are common to most projects and are expected to be tailored to a given situation. And the people performing the process have a significant role in the design and management of the process. Naturally, it requires more skill to make good judgments about how to tailor a process to fit a business and project environment.

PRINCIPLES OF PRODUCT DEVELOPMENT FLOW

Don Reinertsen wrote a widely read book on the principles of product development flow, summarized here:[17]

1. *Economics: Take an economic view.* Many times there is a point of diminishing returns associated with improvements in a process—understanding the economic impact is a critical factor in optimizing a process. For example, it is *generally* best to defer decisions on product features as long as possible; however, some decisions should be made early and should not be significantly deferred because of their economic impact.

   Example: Increasing innovation should not be an end-in-itself and it reaches a point of diminishing returns at some point and begins to impact other proxy variables such as quality.
2. *Queues: actively manage queues.* Agile development processes are based more on a continuous flow process as opposed to heavily plan-driven processes that are more of a large-scale batch process. A continuous flow product development process operates most efficiently when queues are managed.

[17] Don Reinertsen, *The Principles of Product Development Flow* (Redondo Beach, CA: Celeritas Publishing, 2009).

Example: Developing requirements far-in-advance that sit in a queue waiting for development can be inefficient and wasteful because:

- The requirements may change prior to going into development and much of the effort involved in developing the requirements might have been wasted, and/or
- Speculation in the requirements that are done too far into the future can result in erroneous assumptions that make their way into development without being questioned.

3. *Variability: Understand and exploit variability*. Reducing variability will *many times* improve efficiency but that isn't always the case. For example:

- Breaking up large requirements into smaller ones that are of a more uniform size reduces variability and can improve flow, however, at some point further attempts to reduce variability do not have economic value.
- Forecasting errors are a major source of variability . . . we can reduce this variability by forecasting at shorter time horizons.
- Design reuse reduces variability.

4. *Batch size: Reduce batch size*. Large batch sizes tend to cause bottlenecks and inhibit flow. Reducing the batch size by breaking up requirements into small, independent user stories can significantly improve flow.

   Examples of batch size inefficiencies include:

- Project scope: More is taken on in a single project than is truly necessary.
- Project funding: The entire project is conceived and funded as a single large batch proposal.
- Requirements definition: the tendency to define 100% of the requirements before the project starts

5. *WIP constraints: Apply WIP constraints*. Use work in process (WIP) constraints to manage overall flow. For example,

- Control the number of projects taken on at any one time to avoid oversaturating development resources.
- Use specialized resources wisely to maximize their impact on overall flow.

6. *Control flow under uncertainty: Cadence and synchronization*. Reinertsen[18] defines cadence as follows:

> Cadence is the use of a regular predictable rhythm within a process. This rhythm transforms unpredictable events into predictable events. It plays an important role in preventing variability from accumulating in a sequential process . . .

[18] Ibid

Having a repeatable cadence improves the efficiency of the product development process and allows synchronizing a predictable development process with a much more unpredictable flow of requirements. Examples of the use of synchronization include:

- Concurrent development on multiple paths at the same time
- Concurrent testing of multiple subsystems

7. *Fast feedback: Get feedback as fast as possible.* Fast feedback can lower the expected loss by truncating unproductive paths more quickly or raise the expected gain because we can exploit an emergent opportunity by rapidly redirecting resources.

   Fast feedback, combined with selecting appropriate measures of performance, enables rapid learning and ongoing continuous improvement.
8. *Decentralized control: Decentralize control.* The final principle that Reinertsen has identified deals with decentralized control:

> Sophisticated military organizations can provide very advanced models of centrally coordinated, decentralized control. There is an impression that military organizations seek robotic compliance from subordinates to the orders of superiors. In reality, the modern military focuses on responding to a fluid battlefield, rather than executing a predetermined plan. It views war as a domain of inherent uncertainty, where the side that can best exploit uncertainty will win.[19]

SUMMARY OF KEY POINTS

Systems Thinking

1. Systems thinking is very important in order to see agile principles and practices in a holistic sense and in the context of how they fit with the overall business objectives of an enterprise. In the context of agile, *systems thinking* means understanding the principles behind the methodology rather than just focusing on the mechanics of how the methodology works and understanding how those principles interact with the overall project and business environment that they are part of.

Influence of Total Quality Management (TQM)

2. TQM and the thinking behind it revolutionized the quality and competitiveness of the automotive industry. TQM taught us to:

   - *Eliminate defects at the source:* Develop a more systemic approach to designing quality into the process that produces products to eliminate the defects at the source, rather than constantly finding and fixing the same or similar defects over and over again.

[19] Ibid.

- *Recognize the human aspects of quality:* Recognize the human aspects that are essential to build quality, such as engaging people at all levels so that they feel responsibility and ownership for the quality of the overall products they produce and empowering them to recognize and suggest opportunities for improvement in the process.
- *Develop a cross-functional approach:* Break down barriers between departments, eliminate conflicting goals among organizations, and develop a much more integrated cross-functional approach that leads to much higher levels of productivity and efficiency, together with a more collaborative approach to quality and process improvement.
- *Recognize the importance of leadership:* Eliminate traditional command-and-control management in favor of inspirational leadership to empower people and to help them see the higher-level purpose and vision for their work.
- *Strive for ongoing continuous improvement:* Commit to an ongoing continuous improvement effort to constantly find opportunities to improve processes.

Influence of Lean Manufacturing

3. The concept of lean originated in manufacturing and has had a significant impact on many industries. The focus of lean manufacturing is on elimination of waste and improving operational efficiency rather than simply improving quality; however, lean does recognize quality defects as an important form of waste that should be eliminated. Lean taught us:

- *Customer value focus:* Focus on producing customer value and eliminate all unnecessary tasks that do not add value to the customer.
- *Map the value stream:* Map the value stream to understand the process flow and identify any opportunities to eliminate non–value-added steps.
- *Pull:* Use a pull approach rather than a push approach to plan and manage production capacity to meet demand.
- *Flow:* Use the principles of flow, such as just-in-time processing, to optimize the efficiency of the overall process.
- *Respect for people:* Recognize and be sensitive to the human aspects of quality. People need to be respected and properly motivated to develop high-performance teams that produce very-high-quality products.
- *Perfection:* Use a systemic approach to identify the source of waste and defects in a process and use a continuous incremental improvement approach to perfect the process.

These same principles are adaptable to a software development environment.

Principles of Product Development Flow

4. A traditional management approach is heavily based on managing and controlling the *structure* of a project with such things as work breakdown structures, Pert charts, and Gantt charts. An agile project is based on a much more fluid and adaptive process where structure is much less important and managing the flow of items through the process is much more important.

5. Understanding the principles of product development flow is very important to optimize the efficiency of an agile project. The key principles of product development flow outlined by Don Reinertsen are:
   - Economics: Take an economic view.
   - Queues: Actively manage queues.
   - Variability: Understand and exploit variability.
   - Batch size: Reduce batch size.
   - WIP constraints: Apply WIP constraints.
   - Cadence and synchronization: Control flow under uncertainty.
   - Fast feedback: Get feedback as fast as possible.
   - Decentralized control: Decentralize control.

DISCUSSION TOPICS

Systems Thinking

1. Discuss an example of a problematic situation where you might have used systems thinking to analyze the situation and determine an appropriate solution.

Influence of Total Quality Management

2. Discuss an example of a problematic situation where the TQM principles might have been used to improve the overall quality of the work. What were some of the issues that impacted the quality of the overall product or service that the company produced? How could the impact of those issues been reduced or eliminated?
3. How do you think that an understanding of the principles behind Total Quality Management might influence an agile project management approach?

Influence of Lean Manufacturing

4. Discuss an example of a situation where lean thinking could have been used to improve the overall efficiency of the process. What was some of the waste involved in the process? How could it have been reduced or eliminated? What principles of lean did you use to make that assessment?
5. How do you think that an understanding of lean manufacturing principles might influence an agile project management approach?

Principles of Product Development Flow

6. Analyze the following situations and identify the appropriate principles of product development that you might be important to consider in defining an appropriate solution to each:
   - A project is taking much longer than originally anticipated: it was originally planned to take only 6 months, it has now gone on for over 18 months, and there still doesn't seem to be an end in sight.

- Projects are being delayed because users are frequently changing requirements in the middle of the project.
- The quality assurance organization is becoming a bottleneck and scheduled releases are being delayed waiting for QA testing.
- A company has just finished a massive implementation of agile throughout the entire company but senior executives are not satisfied with the results. They feel like they have lost some visibility into projects and are not sure the projects are well-aligned with their business goals.
- An agile development team is repeatedly missing its sprint goals because it has overcommitted the amount of work that can be done in a sprint.
- The company CEO insists on personally making decisions in what he considers to be the most critical projects to the company's success.

PART 3

Making Agile Work for a Business

THERE ARE MANY PRECEDENTS FOR successful implementation of agile principles and practices at a project team level; however, extending the agile principles and practices to large-scale enterprise implementations and integrating with a business environment introduces a number of new challenges, which include:

- Large, complex projects that are commonly found at an enterprise level may require some reinterpretation and adaptation of agile principles and practices.
- Integrating agile principles and practices with higher levels of management typically found at an enterprise level, such as project portfolio management and overall business management can be difficult. However, if an agile implementation is limited to a development process only and does not address integration with these higher-level processes it is not likely to be completely effective.

This section of the book is intended to address these topics and provide an understanding of the key considerations that need to be addressed for:

- Scaling an agile approach to an enterprise level
- Integrating it with a business environment
- Planning and implementing an agile transformation

Chapter 12 – Scaling Agile to an Enterprise Level

Scaling an agile approach to an enterprise level requires some reinterpretation of how the agile principles and practices apply in an enterprise-level environment. It also requires some planning

to address how the agile approach will integrate with higher-level management processes such as project management and project portfolio management that are found at an enterprise level.

Chapter 13 – Adapting an Agile Approach to Fit a Business

Discusses factors associated with the company's business environment, management structure, and culture and values that are likely to impact an enterprise-level agile implementation.

Chapter 14 – Enterprise-level Agile Transformations

This chapter provides some guidelines for planning and managing an enterprise-level agile transformation.

12 Scaling Agile to an Enterprise Level

AS A RESULT OF THE widespread adoption of agile practices, large corporations are beginning to apply agile at an enterprise level. This has introduced some new challenges of how to scale agile principles and practices to an enterprise level and what to do about many of the existing project management office (PMO) practices and other higher-level business management practices that are typical in large enterprises for managing portfolios of projects and products.

> Our experience is that 'core' Agile methods such as Scrum work wonderfully for small project teams addressing straightforward problems in which there is little risk or consequence of failure. However, 'out of the box,' these methods do not give adequate consideration to the risks associated with delivering solutions on larger enterprise projects, and as a result we're seeing organizations investing a lot of effort creating hybrid methodologies combining techniques from many sources.[1]

Dean Leffingwell identifies two primary challenges involved with scaling agile to the enterprise level in his book, *Scaling Software Agility—Best Practices for Large Enterprises*:[2]

1. The first challenge is overcoming the challenges inherent in the methodology. Scaling an agile methodology to an enterprise level requires reinterpreting the values and principles behind the methodology in a much larger context, and typically also requires some adjustments in agile practices to adjust to that context.
2. The second challenge is overcoming the limitations imposed by the enterprise that will otherwise prevent the successful application of new methods. As I've mentioned earlier, implementing an agile approach at an enterprise level can be like plugging an appliance that

[1] Scott Ambler and Mark Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise* (IBM Press) (Upper Saddle River, NJ: Pearson Education, 2012; Kindle Edition), pp. 349–353.
[2] Dean C. Leffingwell, *Scaling Software Agility* (Reading, MA: Addison-Wesley, 2007), p. 87.

requires DC current into an AC outlet; there's a fundamental incompatibility in many cases that requires some adaptation to get an agile approach to work inside of an organization that wasn't designed around being agile.

The first challenge will be discussed in this chapter, and the second challenge will be discussed in the Chapter 13, "Adapting an Agile Approach to Fit a Business."

ENTERPRISE-LEVEL AGILE CHALLENGES

Beyond the factors previously discussed to adapt an agile approach to a company's business environment, there are a number of challenges associated with scaling agile practices to an enterprise level.

Differences in practices

Applying agile development practices at the enterprise level typically involves some adjustments to those practices, as shown in Table 12.1.[3]

TABLE 12.1 Adaptations to Agile Practices at the Enterprise Level

Typical Small Agile Project	Typical Enterprise-Level Implementation
Customer Participation The customer is integral to the team.	The customer may be remote or may not have the skills or time available to participate directly in the agile team. The customer may also consist of a number of different users and stakeholders. A business analyst (BA) on the team many times plays the role of a proxy for the customer, but that BA should ideally be empowered to act on behalf of the customer.
Development Team Organization Developers, product owners, and testers are co-located and not separated by time zones and language barriers.	It is likely that many team members may be in different geographic locations and different time zones and perhaps even speak different languages.
Application Architecture In a small-scale agile project, the application architecture is expected to emerge as the project progresses.	With larger-scale systems, the costs and difficulty of refactoring the design as the project progresses to accommodate changes in the architecture make it essential in many cases to do more upfront architectural planning and design in the project. Large-scale system designs typically require breaking up the design into components, and without having sufficient architecture defined, it becomes impossible to allocate the work to teams.[4]

[3] Ibid., pp. 88–89.
[4] Ibid., p. 204.

TABLE 12.1 (*Continued*)

Typical Small Agile Project	Typical Enterprise-Level Implementation
Requirements Management The agile development effort can take place one story at a time, and the design incrementally evolves over the duration of the project.	In large enterprise-level implementations, this approach doesn't work very well. A more integrated approach may be required to coordinate the development of the stories to ensure that they all really work together to produce releasable functionality that fulfills the business need.
Project Portfolio Management Typical agile projects do not provide a mechanism for higher-level integration to fulfill typical corporate needs for portfolio management of a large set of agile projects.	It can be very difficult to integrate a number of agile projects into a typical enterprise-level project portfolio management approach; however, some level of integration and management is necessary to make portfolio management decisions. This will many times require adopting a hybrid approach to provide the necessary balance of predictability, control, and agility.
Team Organization Ideally, agile teams consist of peer-level developers who take responsibility for their own actions with a minimum of direction. It is intended to be a team of equals with no formally designated technical team leader.	For large-scale development teams, this can be very difficult, if not impossible, to achieve. Many times it is necessary to build teams of more junior-level developers led by a more senior-level tech lead who can provide some level of guidance and direction to the rest of the team.

Reinterpreting agile manifesto values and principles

There is also a need to reinterpret the Agile Manifesto values and principles in a different context at the enterprise level, as shown in Table 12.2.

TABLE 12.2 Agile Manifesto and Enterprise-Level Implementation

Typical Small Agile Project Values	Typical Enterprise-Level Implementation
Tools: "Individuals and interactions over processes and tools"	There is a greater need for tools at an enterprise-level: ■ There is more of a need for a defined process to coordinate and synchronize the work of large projects requiring multiple teams as well as coordinating other activities outside the teams. ■ Tools can become more important at an enterprise level as the scope and complexity of the effort grows.

(*continued*)

TABLE 12.2 (*Continued*)

Typical Small Agile Project Values	Typical Enterprise-Level Implementation
Documentation: "Working software over comprehensive documentation"	■ At an enterprise level, solutions tend to be much more complex, and software is only one part of the overall solution. As a result, some form of additional overall coordination is needed to integrate all the components of the overall solution. ■ At an enterprise level, a solution might also include training, business process changes, a support plan, a marketing and rollout plan, and many other requirements beyond just developing software. All of these may increase the need for some kind of documentation.
Collaboration: "Customer collaboration over contract negotiation"	■ At an enterprise level, there is typically a much broader range of customers and stakeholders to consider and managing expectations can be a lot more challenging. ■ Some form of project charter document may be worthwhile to help manage expectations but it could be defined at a fairly high-level.
Planning: "Responding to change over following a plan"	As the scope and complexity of an effort at an enterprise-level increases, there is typically a need for more planning to: ■ Coordinate the efforts of large projects requiring multiple teams. ■ Synchronize the efforts of development teams with other activities outside the scope of the development effort. ■ Adapt the development effort into higher-level management processes that may be more plan-driven.
Change Control: "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage."	■ Change never comes without consequences, and change control can be valuable for configuration management and validating that any new changes are consistent with other previously developed requirements and assumptions. ■ Done properly, it does not equate to stifling or *preventing* change. It means ensuring that unnecessary change (as ultimately defined by the sponsor) is rejected, but that necessary change is brought into the project with the full awareness of all concerned and that necessary adjustments to designs, plans, timescales, tests, contracts (etc.) are made with a minimum of wider disruption.
Communications: "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."	■ At an enterprise level, because of the focus on the overall solution, rather than just software, the team is typically broader than simply the people who are developing the software. ■ At an enterprise level, the communications strategy must include a broader set of people such as production operations and support that are important stakeholders in the implementation of the solution.

TABLE 12.2 (*Continued*)

Typical Small Agile Project Values	Typical Enterprise-Level Implementation
Progress Measurement: "Working software is the primary measure of progress."	■ At an enterprise level, success or failure is measured more in terms of delivering *real business value* to the users over simply developing functional software. ■ The primary measure of progress should be whatever the Sponsor defines as value. (Think about user training or process change, for example)

ENTERPRISE-LEVEL OBSTACLES TO OVERCOME

In addition to the challenges previously mentioned, there are a number of obstacles that are commonly associated with an enterprise-level agile implementation.

Collaborative and cross-functional approach

One key challenge is developing the collaborative, cross-functional approach that is required for agile. There are two aspects of this challenge:

1. Agile requires breaking down some organizational barriers that may exist and organizing people into dedicated, collaborative teams. Instead of having a QA department that is separate from the development organization and provided direct oversight of the testing process, you might have a QA organization that provided general functional guidance, but the QA testers would normally work as part of a dedicated team and would not be managed by the QA department on a day-to-day basis.
2. Another obstacle is associated with developing a collaborative partnership relationship between the business organization and the development organization. There is a need to develop a level of trust between these two organizations and a spirit of collaborative partnership rather than an arms-length contracting approach and that can be very difficult to achieve in many organizations that have not been accustomed to working that way.

The challenge to the agile project manager is providing strong leadership to help break down these organizational barriers and implement a more cross-functional management approach.

Organizational commitment

In many companies, agile cannot be implemented without some level of cultural change. Implementing it as a development process only and ignoring the need for organizational change will often result in a very limited level of effectiveness. It might be done that way as a first step to show results quickly; but ultimately, achieving the full benefits of agile will probably require some level of organizational change.

Many people make the mistake of assuming that you have to force the whole company to be more agile in order to implement an agile development process. That is not necessarily the case—a company needs to build its culture around whatever makes sense for the primary business that the company is in. Although becoming more agile is a good thing in most companies, it has to be positioned in the context of the company's overall business strategy. *Just becoming agile should not necessarily become an end in itself*.

The challenge an agile project manager may face is helping to plan how an agile development process should be integrated with the company's primary business environment and then helping to lead whatever enterprise-level transformation is needed. Developing the appropriate strategy can be a very challenging role and might involve making compromises between trying to transform the company to become more agile and adapting an agile development project management approach to fit with the company's existing business environment and culture.

Once the overall strategy is determined, implementing that strategy can also be very challenging because it can involve a significant amount of change management for whatever organizational and cultural changes may be required.

Risk and regulatory constraints

There may also be factors in the company's business environment that impose constraints on how far you can go with an agile implementation that need to be taken into consideration. For example, if a company operates in an environment that requires some level of risk and/or regulatory control, it may be necessary to adapt the agile approach to fit that environment but it's not impossible to do that with the right approach and tools. Also, requirements traceability and design control combined with an effective testing approach are usually very important criteria for developing an acceptable approach for meeting regulatory requirements. An agile project management tool can provide a way of satisfying those constraints—the tool can help demonstrate that the process does indeed provide an acceptable level control in those areas. The challenge for an agile project manager is in determining how to blend the right level of control and agility to provide the right balance to the company.

ENTERPRISE-LEVEL IMPLEMENTATION CONSIDERATIONS

There are also a number of enterprise-level implementation considerations that impact how agile projects are implemented at an enterprise level.

Architectural planning and direction

Enterprise-level solutions are typically more complex than a small standalone software application and upfront planning of the design architecture is typically needed for a number of reasons:

- The solution often will require multiple teams, and some sort of architectural direction is needed to define how the work among the teams should be organized and coordinated.

- Because the solutions are typically much larger and more complex, there is much greater risk associated with having to redesign and/or refactor the solution after the design is in progress because the level of effort required may be much larger.
- The solution will also need to integrate with other enterprise-level software and conform to whatever standards the organization uses to ensure that it is interoperable with other applications. That will typically require some planning and design reviews early in the project.

Dean Leffingwell has very accurately identified the need for what he calls *intentional architecture* at the enterprise level:

> For small, Agile teams who can define, develop, and deliver a product or application that does not require much coordination with other components, products, or systems, the basic Agile methods produce excellent results. But what happens when those teams must coordinate their activities as their components integrate into subsystems, which in turn integrate into larger systems? Moreover, re-factoring of these larger scale systems may not be an option because many hundreds of person-years have been invested and the system is already deployed to tens of thousands of users.
>
> "For these systems, the Agile component teams must operate within the context of an intentional architecture, which typically has two characteristics: (1) it is component-based, each component of which can be developed as independently as possible and yet conform to a set of purposeful, systematically defined interfaces; (2) it aligns with the team's core competencies, physical locations, and distribution (if this is not the case, it is likely that the teams will realign themselves thereto!).[5]

Of course, this doesn't necessarily mean that a big upfront design approach is needed for every project—common sense should be used to determine the level of depth that needs to go into upfront architectural planning to reduce the risks and uncertainties involved. For example, if there is significant uncertainty associated with the architecture that would have a high potential risk on the project if the architectural direction is not addressed and resolved early on, it may then necessitate including a special iteration (sometimes called a *spike*) to investigate and resolve that uncertainty. One method that is frequently used is to define and develop a prototype or *slice* of how the ultimate system will be implemented as a proof of concept. That prototype, or proof of concept, can then be used as a reference model by the teams designing and implementing the rest of the system.

Enterprise-level requirements definition and management

In some cases, there is also a need for more upfront planning of requirements at an enterprise level:

- Architecture and requirements are intimately related, and it's impossible to define architecture without some idea of what the requirements are. If there is a need to define the architecture prior

[5] Ibid., p. 190.

to development to reduce the risk, it will probably be essential to define more of the requirements up front in a typical enterprise-level agile project.

- The requirements can be a lot more complex, and more upfront analysis of the requirements may be needed to determine the most appropriate solution and the optimum architecture as well as understanding any interdependencies and interrelationships among the requirements. A technique called *functional decomposition* is often used to break down requirements into a logical organization. Functional decomposition is also a useful way of understanding how the requirements are aligned with supporting the business objectives of the system.
- There are typically a larger number of stakeholders involved in the development of the requirements. For example, a support group will many times have a key role in determining supportability requirements.

There are some significant challenges associated with planning and managing requirements for solutions at an enterprise level:

> Agile's practice of working on a few stories at a time is a wonderful focusing mechanism for the team. But in larger systems, what drives these stories into existence? Who says these are the right stories? Will the summation of all these stories (now in the thousands) actually meet our customer's end-to-end use-case needs? Does our team's Product Owner have clear visibility into stories others are building? Are they likely to affect us? If so, when? And when developing solution sets (large sets of products that must be deployed together and support end-to-end use cases for the user or customers), how do we know that the stories on the table will actually work together to achieve the final objective? Can building an enterprise application, one story at a time, possibly work? Well, perhaps not exactly that way.[6]

However, Michael Hurst, corporate PMO for Harvard Pilgrim Health Care, suggests one clear benefit:

> The most Agile teams use their ability to code quickly and efficiently as a requirements discovery process and avoid the overhead of formal specifications. This practice can work effectively because a small team can write and rewrite code at a rate faster than many organizations can attempt to determine and codify their customer requirements anyway![7]

The key thing to consider is that this is not an all-or-nothing decision of having no requirements at all or having large and unwieldy requirements documents. Good common sense should always be

[6] Leffingwell, p. 190.
[7] Dr. Michael Hurst, personal e-mail, April 20, 2014.

used to determine how much upfront planning and what level of detailed definition should go into the requirements for any project. If the requirements planning and definition effort is done using electronic tools rather than traditional Word documents, it can significantly accelerate the development effort once the development is started. For example, defining a structure to the requirements and organizing them as epics and user stories in an electronic tool makes it much easier to plan, organize, and track development and testing tasks against those requirements.

Release to production

The process for releasing mission-critical applications to production is another important factor that increases the complexity of large, complex enterprise-level applications. Dr. Hurst provided some comments on that from his experience with Harvard Pilgrim Health Care, which is one of the major case studies used in this book:[8]

- We have found in the management of large programs that once the team build passes the acceptance criteria test cases, it is really just the first step in releasing to a complex production environment.
- Ok, my component works locally on the development environment (functional testing), and it works on the nearest neighbor environment (contiguous testing), but does it work in the end-to-end production environment for full User Acceptance Testing UAT)?
- We have found that Kanban process works better than Scrum by the Release Management team since they really need teams to go through a sequence of environments and connections on their way to the full production environment. These stages are highly susceptible to staffing and other resource constraints (test data, test environment updating) that allows only so many things to be in queue at once, but items in queue can be replaced by other things as ones pass through successfully. Perfect production line Kanban.

The whole process of releasing applications to production in an enterprise environment can require a lot of coordination outside of the agile team, which is a key area of value-added that a project manager can provide. The challenges associated with this are typically referred to as DevOps:[9]

[8] Ibid.
[9] Blog, "DevOps: What It Is, Why It Exists and Why It's Indispensable," posted by Luke Kanies, August 23, 2011, http://readwrite.com/2011/08/23/devops-what-it-is-why-it-exist#awesm=~oBYtsF1U5Vw3bm.

> In my experience in operations there's always been a difference in perspective between Dev and Ops, but it's always been more of an impediment than a benefit. The common goal should be getting apps deployed as quickly, safely, and efficiently as possible, but each group instead has a more short-term priority not necessarily related to the results the business is looking for. Lee Thompson (formerly of E*TRADE, now of DTO Solutions) coined the term wall of confusion to describe the apparent inability for development and operations teams to communicate around a common goal, and this wall of confusion is a critical barrier to effective teamwork . . .
>
> Ideally, companies work out a system in which whoever makes the mistake pays the price. The reason Ops is so often scared of Dev deploying is that Dev doesn't really care how secure their apps are, how hard they are to deploy, how hard they are to keep running or how many times you have to restart it, because Ops pays the price for those mistakes, not Dev. In most organizations the mandate of a developer is merely to produce a piece of software that worked on a workstation—if it worked on your workstation and you can't make it work in production, it's Operations' fault if they can't get that to thousands of machines all around the world . . .
>
> Google is a great example in switching up that process. When they deploy new applications, the developers carry the pagers until the stop going off—only when they stop getting outage alerts does operations take over the operational running of a system.

ENTERPRISE-LEVEL MANAGEMENT PRACTICES

There are different levels of management that typically come into play in large, complex enterprise-level projects. Some of these levels of management are shown in Figure 12.1.

There are three primary challenges here:

1. Integrating the efforts of multiple teams from a development perspective
2. Aligning the efforts of all teams with the business objectives of the organization
3. Coordinating with other related efforts outside of the project team and providing tracking and reporting to management

All three of those are very significant challenges, and there is much too learn in all three of those areas. Many aspects of the knowledge of how to make agile work at a team level is relatively mature; however, the knowledge of what needs to be done to scale agile to an enterprise level is far less mature. This is also an area where it becomes essential to figure out how to integrate agile principles and practices with traditional plan-driven principles and practices in the right proportions to fit the situation.

Scrum-of-scrums approach

At a minimum, for large projects that require more than one team, some kind of mechanism is needed to coordinate and synchronize the work among individual teams. The *Scrum-of-Scrums* approach is

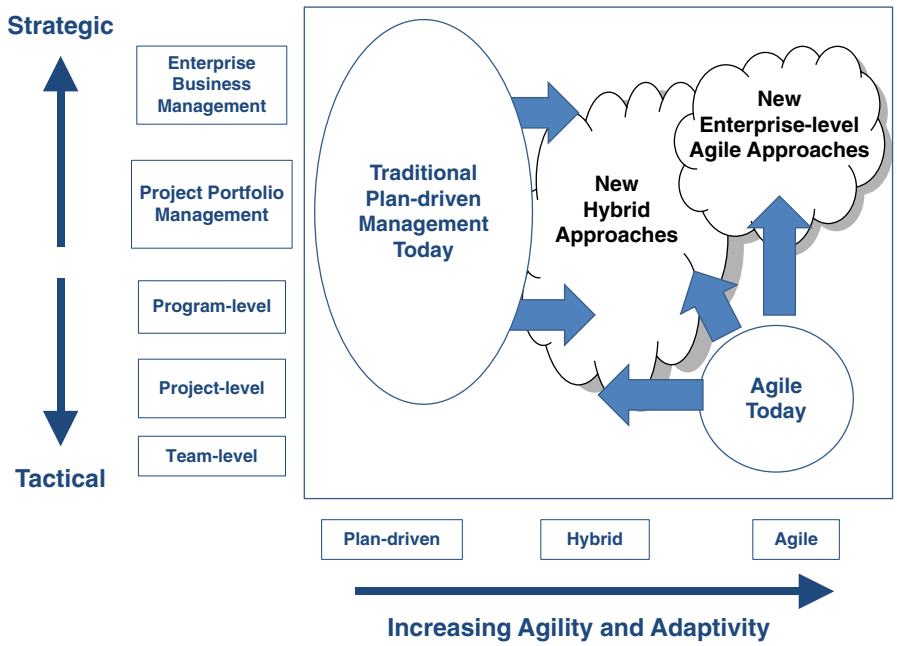


FIGURE 12.1 Typical enterprise levels of management

one way to fill that need. When multiple Scrum teams are engaged in a project, each team does its normal, individual, daily standup meeting to discuss items within the scope of that team's own work, and each team sends a representative(s) to the Scrum-of-Scrum meetings to provide a mechanism for coordination and collaboration across different teams.

> The scrum-of-scrums meeting is an important technique in scaling Scrum to large project teams. These meetings allow clusters of teams to discuss their work, focusing especially on areas of overlap and integration . . . Each team would then designate one person to also attend a scrum-of-scrums meeting. The decision of who to send should belong to the team . . .
>
> Being chosen to attend the scrum-of-scrums meeting is not a life sentence. The attendees should change over the course of a typical project. The team should choose its representative based on who will be in the best position to understand and comment on the issues most likely to arise at that time during a project.[10]

The frequency of the meetings should be determined by the teams depending on the nature of the project and the amount of cross-team communication and coordination required; however, the

[10]Mike Cohn, "Advice on Conducting the Scrum of Scrums Meeting," Scrum Alliance, May 7, 2007, http://www.scrumalliance.org/articles/46-advice-on-conducting-the-scrum-of-scrums-meeting.

meetings may not need to be daily. The organization of a Scrum-of-Scrums meeting follows the same general format as a daily Scrum meeting for one of the individual teams. It should be short (typically no more than 15 minutes), and it is focused on the same types of questions as the daily stand-up meeting for individual teams, as explained by Mike Cohn, president of Mountain Goat Software:

> Because the scrum-of-scrums meetings may not be daily and because one person is there representing his or her entire team, these three questions need to be rephrased a bit. I also find it beneficial to add a fourth question:
>
> 1. What has your team done since we last met?
> 2. What will your team do before we meet again?
> 3. Is anything slowing your team down or getting in their way?
> 4. Are you about to put something in another team's way?[11]

The Scrum-of-Scrums approach is a good mechanism for coordinating the work of multiple teams in a project, but it clearly has its limitations:

- It is highly dependent on the maturity level of the individual teams to be totally self-organizing, not only within each individual team but across all of the teams in the project.
- It is typically limited to coordinating the activities of development teams and is not typically used for coordinating other activities outside the scope of the development teams. For that reason, it is not really intended to be an overall project management approach. Overall, project management is more the domain of the product owner(s) than it is the domain of the Scrum team and the people who may participate in the Scrum-of-Scrums meetings.

Figure 12.2 shows how the Scrum-of-Scrums approach works.

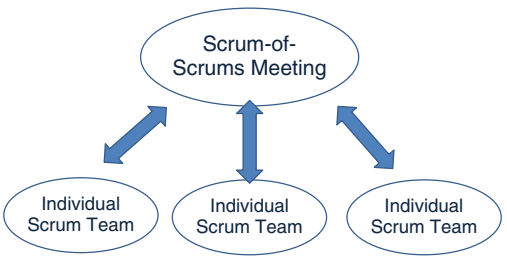


FIGURE 12.2 Scrum-of-Scrums meeting approach

[11] Ibid.

A representative of each individual Scrum team participates in the Scrum-of-Scrums meetings to represent the interests of each Scrum team and coordinate activities:

- The representative may or may not be the Scrum Master.
- The role may be rotated among different team members depending on the need.

Project/program management approach

At an enterprise level, a number of factors contribute to the need for some level of project/ program management in an agile project:

Multiple Distributed Teams

Because of the size of enterprise-level projects, there will frequently be a need for multiple teams. In many cases, those teams may be distributed in different locations and cannot be easily collocated. That will require some level of coordination and communications among and across those teams to ensure that those efforts are well synchronized and consistent with producing the intended business results. The Scrum-of-Scrums approach may be a partial solution to that need, but it is not really intended to provide to be a substitute for project management.

Integration with Higher-level Business Goals

A major source of value-added that a project manager can provide at the enterprise-level is integration with the company's business objectives. At the team level, that role is typically provided by a product owner. However, at an enterprise level, the workload associated with that function might easily justify a project manager. In some cases, the project manager might facilitate a group of product owners or other business stakeholders who act as a steering group for the overall project/program.

Coordination with Other Organizations and Stakeholders

Because large, enterprise-level solutions can have a very broad impact, there is typically a significant need for coordination with other organizations and stakeholders who might be outside of the direct day-to-day project team.

Management of Other Related Activities

At an enterprise level, a number of project-level activities might be outside of the individual project team, such as planning and managing release to production, business process changes, user training, and support requirements.

The role of a project management office (PMO)

A Project Management Office (PMO) in a company typically has several major roles.

Project/Product Portfolio Management

The role of a PMO is primarily to act on behalf of the appropriate business managers to manage the implementation of the company's project/product portfolio management strategy. As a result, the role of the PMO will vary, depending on the level of rigor and control behind the project/product portfolio management approach:

- In a more traditional project/product portfolio management approach, investment decisions to invest in projects will typically require at least a high-level estimate of the costs and schedules of those projects so that they can be intelligently evaluated against other alternatives.
- In a more agile environment, a much more dynamic approach with less financial rigor behind it might be used.

Progress Tracking and Reporting

Once a project has been initiated, at least some minimal form of tracking of progress is needed to determine if the projects are, in fact, actually producing the desired business results and return on investment. The actual level of tracking and control should also be commensurate with the level of rigor in the overall project/product portfolio management approach to determine if the projects/products are really fulfilling their goals:

- In a more traditional environment, there is likely to be a much more rigorous and detailed approach to tracking progress against specific goals and milestones, and the PMO will likely play a significant role in consolidating and validating the reporting information.
- In a more agile approach, the approach may be less rigorous and might put more responsibility on the individual project teams for reporting their own progress. The PMO might play more of a facilitative role and less of a controlling role.

Project Methodology

Another typical role of a PMO is to provide a focal point for sponsoring and managing project methodologies and standards used by the organization:

- In a traditional PMO, the methodologies might be fairly rigidly defined and the PMO might have a responsibility for ensuring compliance with the methodology process requirements, such as completion of required documentation and phase-gate reviews.
- In a more agile PMO, there is likely to be a much higher level of flexibility and adaptivity delegated to the project teams in how the methodology is implemented, and the PMO may play more of a consultative and supporting role to provide training to the project teams as needed to help make them successful.

Project/product portfolio management

Many companies have some form of project/portfolio management approach that is designed to manage the return on investment from their projects. They may also have a PMO structure in place to provide overall management of those projects and to do some form of resource and capacity planning. There's a misconception that an agile development approach is totally inconsistent and incompatible with that kind of PMO management scheme—I don't believe that is the case. There are a range of different approaches that can be adapted to the company's business strategy.

Traditional Financial Portfolio Management Approach

One factor in determining the project/product portfolio management strategy is the level of financial rigor required in making project/product portfolio management decisions. If it is expected that product/product portfolio management decisions will be made on the basis of quantifiable data such as ROI or IRR, it will likely slant this toward a more traditional project/product portfolio management approach. It would be difficult, if not impossible, to do in a true agile approach because there typically just isn't that much information available to support that kind of analysis upfront. In an environment with very high levels of uncertainty, it might be impractical to try to take that kind of approach.

Agile Portfolio Management Approach

A more agile approach is described in the Valpak case study later in this book. Valpak created high-level epics for each of their major business initiatives and used a high-level team of senior executives to plan and prioritize these high-level initiatives similar to the way you might plan and prioritize product backlog items at a project level. The advantages of this approach are that it is very dynamic and can be shifted easily to adapt to different business conditions and priorities.

Lean Startup Approach

An even more agile approach is the *lean startup* approach that was described in the previous section. The lean startup approach is really well-suited for companies that have a very high level of uncertainty associated with their business initiatives and want to take more of an incremental approach to trying out initiatives to see how they work before making a major commitment to any of them.

The key thing to recognize is that it is also not an all or nothing decision to have no management at all or totally oppressive over control with rigorous financial analysis of projects. The alternatives already described provide different levels of control versus agility, depending on the level of uncertainty in the company's business, the company's culture, and other factors.

SUMMARY OF KEY POINTS

Implementation of an agile approach at an enterprise level requires reinterpreting some of the agile values and principles in a very different context. It may also require adapting some of the agile practices to work in a somewhat different environment. The following is a summary of some of the key differences:

1. **Enterprise-Level Agile Challenges**

   There is a need to reinterpret the agile values and principles at an enterprise level in a different context:

   - Implementing agile values and principles at an enterprise level requires a focus on overall *solutions,* not just *software,* and implementation of those solutions might typically require coordination with other activities outside of the direct realm of software development. There must be a plan for how to achieve that coordination that goes outside of the boundaries of the immediate software development team.
   - Many times at an enterprise level there are a number of different stakeholders who have input into the development of a solution. That will require an approach for ensuring that the inputs of those stakeholders are effectively integrated into the project as it progresses.

2. **Enterprise-Level Obstacles to Overcome**

   There are a number of potential obstacles to overcome at an enterprise-level to get an agile approach to work. One of the biggest challenges is to develop a collaborative and cross-functional approach, and that is very critical to successfully achieving enterprise-level agility.

3. **Enterprise-Level Implementation Considerations**

   At an enterprise level,

   - Architectural considerations of how an application interacts with other parts of the architecture can become very significant and may require more upfront planning to ensure that any interdependencies and constraints that must be observed in the architecture are understood and incorporated into the design of the software.
   - The requirements can be much more complex, and more upfront analysis of the requirements may be needed to determine the most appropriate solution and the optimum architecture.
   - The process for releasing projects to production at an enterprise level will typically impose some constraints that must be considered.

4. **Enterprise-Level Management Practices**

   As projects are scaled to an enterprise level, a number of higher-level management considerations beyond the level of individual teams come into play.

There are three primary challenges here:

1. Integrating the efforts of multiple teams from a development perspective
2. Aligning the efforts of all teams with the business objectives of the organization
3. Coordinating with other related efforts outside of the project team and providing tracking and reporting to management

DISCUSSION TOPICS

Enterprise-Level Agile Challenges and Obstacles

1. What do you think is the most significant difference you will encounter in a typical enterprise-level agile project? Why?
2. What do you think is the most important obstacle to overcome in implementing an enterprise-level agile transformation? Why?

Enterprise-Level Management Practices

3. What are the limitations in a Scrum-of-Scrum approach?
4. How is the role of a Project Management Office (PMO) different in an agile environment?
5. What is the lean startup approach? What value does it provide? Where would it be most useful?

13 Adapting an Agile Approach to Fit a Business

THE MISTAKE MANY COMPANIES seem to make is in failing to see the big picture of how an agile development process fits into their overall business strategy. They treat the development process as if it were in a cocoon that can be totally isolated from the higher levels of management that it is part of. There are hundreds of books written about how to optimize every aspect of an agile development process at a team level inside of that cocoon, but much less is written about how to integrate an agile development process with a company's overall business environment.

Many people seem to see agile as a silver bullet; it seems to be taken for granted that a standard agile development process will work in all business environments without modification, and the company will somehow adapt their business to work with the agile development process. In many cases, it should work in the other direction—rather than adapting their business environment to fit a development process, the right approach in many cases is to adapt the development process to fit the company's business environment. Of course, the exception to that is the case where you know that the business environment itself is dysfunctional, needs to be improved, and/or would benefit from becoming more agile. That is a situation where it might make sense to adapt the business environment to fit better with the development process. However, many times more of a compromise or an incremental change is required rather than a radical transformation because of the difficulty of significantly changing the company's overall business environment.

THE IMPACT OF DIFFERENT BUSINESS ENVIRONMENTS ON AGILE

One of the biggest factors that causes a misalignment is the difference between a product-based business management model and a more project-based business management model and how those models are funded. Here are a few example scenarios.

Product-oriented companies

Agile aligns most easily in companies that are in the business of developing products for external sale. This would include companies whose primary business is selling:

- Software products (e.g., Oracle, Microsoft, Intuit)
- Services based on software (e.g., Facebook, Twitter, etc.)
- Hardware products that contain a substantial amount of software (e.g., Cisco, Garmin)

In this type of company, there is a natural alignment between the principles and practices of an agile development process and the company's overall business objectives and culture. The business success of this type of company is critically dependent on developing excellent, leading-edge products that dominate their marketplace and their business success depends on getting those products to market quickly and efficiently with very high levels of quality. Those values are consistent with an agile development process.

A pure agile approach is very well suited for these companies. The key thing that differentiates these companies is that the product development process is tightly and directly connected with their business objectives. These companies need to rapidly adapt their products to market needs, which is typically an ongoing and incremental development process. What usually happens in these companies is that:

- The company enhances their product(s) continuously to meet market needs and competition.
- Rather than having a single project to complete that has a clearly-defined beginning and end, there is a budget and a team of people dedicated to ongoing improvement and support of the company's product(s).

In this type of environment, managing costs and schedules may not be the highest priority; the costs for the development team are relatively fixed, and responding to change and getting products to market quickly to be competitive is often a much higher priority.

- The budgeted costs are planned to provide a given level of capacity for ongoing development, enhancement, and support of the product(s).
- The primary decision is how to allocate capacity to feature development and support for the product(s), which lends itself very well to a pure agile development process.
- The focus is more on making product(s) competitive than on managing costs. The key management challenge is: Are we adding the right features fast enough to keep up with what the market wants and to stay ahead of any competition?

At the highest level, a company like Intuit would probably do some product portfolio management to allocate funding to the various products it sells in order to maximize the return on that investment. A certain amount of investment funding would be allocated to each of the products for ongoing

development and support. An agile development approach can then be used to prioritize the features and capabilities to be developed that would add the most value.

In many of these companies, new technologies such as software-as-a-service (SAAS) and cloud computing have made it possible to get products to market much more rapidly, which has further accelerated the importance of using more agile development processes in these companies.

Technology-enabled businesses

Another type of company that agile aligns well in is a company whose business may not directly involve developing products but where technology plays a very critical and important role in the company's business success. An example would be companies like Amazon.com or Expedia.com—they don't develop products per se; they provide online services to customers. However, their business success is enabled and critically dependent on Internet technology that underpins the business. Where would Amazon.com or Expedia.com be without having superior Internet technology that powers their business and provides a competitive advantage to them against other potential competitors? Because the underlying technology plays such a critical role in their business and it's a very competitive and dynamic marketplace, an agile development process also works well in this type of company.

At the highest level, the company would probably allocate a certain amount of investment funding for the ongoing development and support of the underlying Internet technology that is essential to their business, and the model for allocating that funding to new technology and features would likely be very similar to a product-oriented model.

Project-oriented businesses

Where it becomes more difficult to make an agile development approach work is a project-oriented business, where there is not such a strong natural alignment of an agile product development process and the factors that drive the company's primary business success. These companies do not sell software as a product or service but might need a significant amount of software application development to support their own internal business operations. They provide this support through their internal IT organization.

Examples of companies in that category would be companies that sell products and services that only indirectly use information technology in their business operations for improving operational efficiency, and so on. For example, Harvard Pilgrim Health Care is a large case study discussed later in this book. Technology helps Harvard Pilgrim Health Care to more efficiently process huge numbers of healthcare transactions, but they're not a product-oriented company like Intuit or a heavily technology-enabled company like Amazon.com. They do internal company projects to improve operational efficiency and provide higher levels of service to their customers.

The key thing that differentiates these companies is that the software development effort is less directly connected with their business objectives. In these companies, management of costs

and schedules is typically much more important. There is a budget and resources established for IT; however, that budget isn't allocated to a particular product. There is likely to be considerable pressure to limit that expenditure as much as possible and to make sure it achieves the desired business objectives. At a high level, what typically happens in this environment is:

- Business groups propose projects to satisfy their operational needs. They usually justify them based on internal factors such as improved productivity, operational efficiency, and cost reduction rather than direct competition with external products. As a result, projects tend to have a beginning and an end and some well-defined objectives they are expected to accomplish.
- There is some kind of project/portfolio management approach that prioritizes those potential projects and determines how the IT budget will be best spent to meet the company's overall business needs. That process typically needs at least a rough estimate of costs and schedules for the potential projects to prioritize the effort and to allocate resources.
- Once resources are allocated to projects and some expectations have been established about the costs and schedule of the projects, some form of tracking and reporting is typically needed to monitor if the projects are on track to fulfill their objectives within the assigned budget.
- The focus is usually much more on managing costs. Project funding should be spent only to the extent that the investments in projects produce an acceptable return in terms of improved productivity, operational efficiency, reduced costs, and so on. The key management challenge is: Are we investing wisely in projects that will improve our business, and are we getting an acceptable return from those investments?

Hybrid business model

Companies in this category don't develop products for external sale, but instead develop products that leverage other revenue-producing products and services. An example of this kind of company is a bank that offers free online services to their customers, such as web-based online checking account access. These services are typically offered to customers at no charge and/or may be heavily subsidized because they are designed to leverage other revenue-producing products or services such as banking services. This type of company shares some of the characteristics of both of the other two categories:

- Although the company (e.g., the bank) may not sell software directly, the software may have a critical impact on the firm's ability to leverage its primary products or services. There is practically always ongoing software development to remain competitive, similar to a company that focuses on software as a product. For example, online bill payment is a common feature that many banks offer, and any bank that does not have that capability might not be very competitive with banks that do.
- These companies don't invest in developing products unless they can produce an acceptable return in terms of leveraged revenue or other factors. They typically have some kind of project portfolio

management approach where potential projects are prioritized and managed to ensure that the company selects the best mix of projects to align with their business strategy and that the projects generate a reasonable return for their investment.

The decision process in these companies is similar to the decision process in companies that develop projects for internal use only; however, in these companies, customer-facing software projects that leverage the company's primary revenue-producing products and services may compete for resources and budget against internal IT projects that are designed to improve operational efficiency. An integrated approach is needed that ensures that those investments are consistent and complementary to each other.

Adapting an agile approach to a business

Some people will say that agile only works in product-oriented companies and can't work *at all* in a project-oriented environment. I don't believe that is the case—it can be a lot more difficult, it may require some adaptation, and it may take more skill, but it definitely can be done. Some of the challenges in a project-oriented company will center on how projects are funded:

1. Some level of upfront planning and financial analysis might be necessary to support a project portfolio management decision process to determine which projects will provide the most optimum return to the company.
2. Some level of tracking might be needed to validate that the projects are really providing the expected return once they are approved.

In these companies, some compromises might need to be made to make an agile approach work. If a company insists on a very rigorous level of financial analysis (NPV, ROI, IRR, etc.) to justify and track project investments, it will naturally make it much more difficult to implement a pure agile development approach. There are two potential compromises that can be made in this kind of environment:

- The company may need to adopt a less rigorous and more dynamic approach to doing financial analysis and tracking of projects. Valpak (which is another case study in this book) is an example of a company that uses a very dynamic approach for project portfolio management.
- It also may be necessary to adopt more of a hybrid model for projects than a pure agile approach (see the discussion in this book on the Managed agile Development approach) to provide a plan-driven wrapper around agile projects. However, that plan-driven wrapper can be as thick or thin as you want it to be—it need not be a very rigorous and highly controlled model.

Even in companies that develop products for external sale, there is a need for some level of overall management, but the management approach for managing investments in an ongoing product development environment is very different than managing investments in internal IT projects. The key point

is that in all of these categories, there may be a need to blend some level of traditional plan-driven management discipline to manage costs and schedules in order to manage return on investment with an agile development approach in the right proportions to fit the situation.

TYPICAL LEVELS OF MANAGEMENT

There are several levels of management that are found to some extent in all of these companies as shown in Figure 12.1.

Overall business management level

At the highest level, the overall business management level, the focus is on:

- Determining the company's overall business and marketing strategy;
- Developing a corporate culture and values that are consistent with that strategy; and
- Providing overall leadership to build cross-functional commitment to execution of that strategy.

It's obvious in companies whose primary business is to develop products for external sale that there is a direct relationship between the high-level business management approach and an agile development approach, and it's very easy to make a strong case for integrating the two. It's also generally easier to implement, because everyone in the organization shares a common focus on making the products that the company produces successful. However, in a company where that is not the case, that can be much more difficult:

- The relationship between the company's overall business goals and an agile development approach might be much more indirect. There are typically many more organizational challenges to resolve, and it can be more difficult to build the cross-functional synergy that is needed to develop an effective agile development approach.
- There also may not be a natural alignment between the culture and values of the company and an agile development approach. For example, in a company whose business success requires a culture with a significant focus on operational excellence, it may be unrealistic and inappropriate to force the company to change its overall business culture to adapt to an agile development approach.

Overall Management Approach

Many companies are deeply rooted in a traditional business management approach. The very structure of the organization may be a significant obstacle to overcome, and that can put a severe strain on the organization in trying to become more agile. Solving this problem calls for new styles of management that many of these companies are not well-prepared to adopt because they typically have hierarchical management structures that are not consistent with an agile development approach. Jurgen Appelo

captured the essence of this problem in his book *Management 3.0—Leading agile Developers, Developing Agile Leaders*:[1]

> Some people call it scientific management, whereas others call it command and control. But the basic idea is the same: An organization is designed and managed in a top-down fashion, and power is in the hands of the few . . .
>
> Some people realized that Management 1.0 doesn't work well out of the box, so they created numerous add-on models and services with a semi-scientific status, like the Balanced Scorecard, Six Sigma, Theory of Constraints, and Total Quality Management. Being add-ons to Management 1.0, these Management 2.0 models assume that organizations are managed from the top, and they help those at the top to better 'design' their organizations. Sometimes it works; sometimes it doesn't . . .
>
> Management 2.0 is just Management 1.0 with a great number of add-ons to ease the problems of an old system. But the architecture of Management 2.0 is still the same outdated hierarchy . . .
>
> One important insight is that all organizations are networks. People may draw their organizations as hierarchies, but that doesn't change that they are actually networks. Second, social complexity shows us that management is primarily about people and their relationships, not about departments and profits.
>
> Many of us already knew that 'leadership' is just a trendy name for managers doing the right thing and doing things right. But complexity thinking adds a new dimension to our existing vocabulary. It makes us realize that we should see our organizations as living systems, not as machines . . .
>
> We have to replace assumptions of hierarchies with networks because the 21st Century is the Age of Complexity.

That is the essence of what Management 3.0 is about. To fully achieve high performance results with an agile development approach, we may need to change our thinking about how organizations are managed. Naturally, that is not going to happen overnight, and a transitional approach might be necessary to get there.

Lean Startup Approach

The *lean startup* concept is a very agile approach can be applied at the overall business management level to the development and planning of strategic business initiatives. It was developed by Eric Ries

[1] Jurgen Appelo, *Management 3.0—Leading Agile Developers, Developing Agile Leaders* (Reading, MA: Addison-Wesley, 2011).

and published in his book in 2011.[2] The simplest way to describe lean startup is an agile approach for developing and managing a very dynamic and successful overall *business strategy*. The word *startup* may be misleading—the approach captures the essence of a very successful, entrepreneurial startup company, but it can be applied to any size company at any stage of maturity. It is not limited to companies that are in the startup phase, and can be used to rejuvenate larger, established companies by infusing the mentality of a successful startup company into the environment.

Eric Ries developed the lean startup concept based on lessons learned from failed attempts of his own to start a business or launch new products. He characterized the root cause of the failures of There, Inc. and Catalyst Recruiting: "It was working forward from the technology instead of working backward from the business results you're trying to achieve."[3] The idea behind the lean startup approach is that many business ventures or new product launches fail because of some faulty assumptions about what the market wants. Instead of recognizing and validating the uncertainty behind those assumptions, companies many times create elaborate business strategies and expensive product development efforts based on some very faulty assumptions that are never fully tested and validated. They don't discover that some of the underlying assumptions behind the business strategy were wrong until an expensive development effort is completed, and by that time it is too late to make a significant change.

The lean startup approach starts with a hypothesis of what the market wants and, through a series of experiments, attempts to progressively validate and refine that hypothesis. The difference is as follows:

- In a traditional product development effort, the business strategy and planning effort and the product development effort are typically sequential. There is an assumption that the business strategy behind the product is complete and is a sound strategy when the development starts, and only details of the requirements need to be resolved.
- In a lean startup approach, the business strategy and planning effort are more concurrent with the development effort. When the development effort starts, the business strategy is recognized as only a hypothesis that needs to be tested, validated, and refined as the development effort progresses.

The lean startup approach can be a very effective way to develop and validate a business strategy for new products in very uncertain markets. It is most effective when there is a high risk associated with a business strategy being successful because of the uncertainty of market acceptance. Here's how Joe Zulli, CTO at Savings.com defines lean startup:

> Lean Startup is a scientific and systematic approach to turning an idea, or vision, into a viable business. One way to think of it is as the scientific method for start-ups. Most

[2]Eric Ries, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses* (New York: Crown Publishing, 2011).
[3]Wade Roush, "Eric Ries, the Face of the Lean Startup Movement, on How a Once-Insane Idea Went Mainstream," *Xconomy* (July 6, 2011).

> recent scientific breakthroughs were achieved not as one singular stroke of genius, but rather as a series of smaller experiments to prove or disprove a single hypothesis. The beauty of the scientific method is that you always win. Even when your hypothesis is disproven, you still come out better off because you've gained valuable knowledge that will make your next hypothesis that much better. Repeat-process enough times, and you will have the breakthrough you were hoping for (assuming you don't give up first). In Lean Startup, we believe the same concept can be applied to business. After all, what really is a 'vision' other than a collection of hypotheses? If you can break a vision down into a prioritized set of assumptions, then efficiently test those assumptions, you will have much more success, in much less time, for much less money, than if you just build everything whole-hog without testing your assumptions along the way.[4]

The lean startup approach forces us to reinterpret several of the basic Agile Manifesto values and principles such as "Working software is the primary measure of progress" in a broader context:

> Above all, Lean Startup is a redefinition of what it means to make progress. The Agile Manifesto states that 'Working software is the primary measure of progress.' Lean Startup, on the other hand, defines the primary measurement of progress as validated learning. After all, what good is working software if it doesn't move you forward as a business by making your customers happy? Sure, you may have thousands of lines of beautiful code, but did you really make progress if your company is no better off? Probably not. When you follow this new definition of progress, you stop worrying about 'velocity' and how much code gets delivered, and start focusing on how you can learn the most the fastest. Sometimes a simple phone interview with a half a dozen prospective customers will get you a lot closer to where you need to be as a company than writing half a million lines of code.[5]

Enterprise product/project portfolio management level

At the enterprise product/project portfolio management level, the challenge is allocating budget and resources to products and/or projects to optimize the return on investment, as well as tracking performance against planned budget allocations to see if the selected projects are providing the expected return.

- In a pure product development company, there may be a mix of different products that need to be managed and optimized. For example, at a high level, Microsoft has to determine how much

[4] Blog, "Agile/Lean at Savings.com: Interview with Joe Zulli, CTO," posted by Stephanie Stewart, October 4, 2012, http://iamagile.com/2012/10/04/agile-lean-savings-com-joe-zulli-cto/.
[5] Ibid.

budget to allocate to ongoing development and support of Microsoft Office versus other products such as Microsoft Visual Studio based on some assumptions as to where it will get the greatest return for that investment. Once the investment decision is made, the company will want to manage the ROI to determine if the investment in new features and enhancement for Microsoft Office paid off in terms of continued revenue growth in that product.

- In either of the other two categories (IT Projects for Internal Use and Hybrid Business Model), it may be more a question of managing a portfolio of *projects* rather than a portfolio of *products,* but the management challenges are similar. In those environments, the challenge is how to invest in a portfolio of projects to get the most leverage from that investment either in terms of improving internal operational efficiency or providing new and enhanced capabilities to external customers that drive revenue from the company's primary products and services.

It can be difficult to integrate an agile development process with an enterprise product/project portfolio management approach, because at the project level, some level of a plan-driven approach may be necessary to at least develop a high-level estimate of costs and schedules to evaluate alternative product/project investments. Also, some form of tracking may also be needed to monitor that the investment is yielding an acceptable return.

Achieving that integration may require two things:

1. Putting some kind of plan-driven wrapper around an agile development approach to provide whatever level of cost and schedule management is desired to support enterprise-level product/project portfolio management decisions.
2. Some tools to roll up data for tracking results across a portfolio of products or projects. Many of these tools also provide capabilities for capacity planning and modeling what-if decisions to evaluate different resource planning scenarios.

In my experience, many companies that implement an agile development process abandon or neglect this level of portfolio management because it can be very difficult to integrate with an agile development approach. That typically results in two things:

1. Shoot-from-the-hip judgments about how to invest and manage a portfolio of agile products and projects, which may not be very effective
2. Very limited or no capacity planning to allocate resources to projects, and constant juggling of resources among projects, which can destroy the effectiveness of an agile development process

Implementing this level of management with an agile development approach can be somewhat difficult to do, but it is very possible with the right tools and with some level of adaptation of the agile development approach. Here are a couple of key points:

1. Having the right tools is essential to make this work at the enterprise level. A company that uses very simple paper-based story boards for their agile teams, without tools that provide the

ability to roll up data across projects, will find it very difficult, if not impossible, to do project portfolio management effectively.

2. It may require a hybrid approach that combines an agile development approach with a plan-driven wrapper to provide some level of predictability and visibility to estimate and manage project costs and schedules. Attempting to do enterprise project portfolio management without some kind of cost and schedule estimates will be difficult, but that is not an all-or-nothing decision. The plan-driven wrapper can be as thick or thin as you want it to be. There are trade-offs to be made when selecting an approach to provide the right balance of agility with predictability and visibility. An example of a hybrid approach that can be tailored to fit a business environment is given in Chapter 16, "Managed Agile Development Framework."

Product management level

The Product Management level is primarily found in companies that develop products for external sale and companies that develop products to leverage other revenue-producing products and services. It might also be found, to a more limited extent, in companies that develop IT projects for internal use if the internal IT applications are large enough and significant enough to warrant some level of ongoing product management. In those companies, this role might typically be performed by a business analyst rather than a product manager, but that's primarily a change in title only; the skills required are essentially the same. The product management challenge is to prioritize the features in a given product to maximize the success of the product and, of course, to validate those features with customers to determine if they really have the expected impact.

For companies that do product management of products for external sale or ongoing development of products for internal use, this level is consistent with an agile development approach. The product backlog in an agile development approach is ideal for managing this kind of effort. However, there still may be a need for more of a hybrid development approach if there is any need for higher levels of predictability in managing the costs and schedules associated with the effort and/or to do product/project portfolio management to manage competing investments in multiple projects or products.

Project management level

In a typical business environment, the next level down would be management of individual projects. This is where the traditional project management practices associated with managing scope, costs, and schedules come into play. The level of emphasis on that kind of project management will naturally vary, depending on the relative importance of managing scope, costs, and schedules in the company's overall strategy:

- *This level may not be found at all in product development companies.* In that environment, rather than breaking up an overall product development effort into discrete projects that each

have a beginning and an end, the company may treat the effort as more of a continuous, ongoing product development process that may not require a very rigorous project management approach. However, even in product-oriented companies, it might make sense to apply some normal project management discipline to product development projects or programs.

- *Managing costs and schedules to support the company's overall project portfolio management approach requires more focus on project management*. This is especially true in the other two environments where product development is not a major focus. However, even in these environments, a hybrid approach rather than a purely plan-driven, waterfall-style approach might also be most appropriate.

CORPORATE CULTURE AND VALUES

Cultural obstacles are well known as some of the most difficult issues to overcome in an enterprise-level agile transformation. However, there is an idea that you should adapt the company's business environment and culture to fit the agile development process, and that is not necessarily a good idea. The overall culture and values of the company should be defined primarily by the success factors that are needed for the markets the business operates in. The culture and values associated with implementing an agile development approach need to be understood in that larger context, but some of the values that are essential for agile, such as respect for people, are likely to be common to some extent to all business environments.

The importance of corporate culture and values

A corporate culture and values that are well-defined and aligned with the company's business strategy can be a powerful unifying force in making good companies great. Stephen Covey summed this up very well in Ann Rhoades' book, *Built on Values—Creating an Enviable Culture that Outperforms the Competition:*[6]

> In order to be successful in a volatile world, you must unleash the goodwill and creativity of your people. You must organize your culture in a way that will help your people achieve great things without constant supervision from above. Set this up right, and people will astonish you regularly with their great ideas and ability to take your organization to a higher level . . .
>
> Companies that have significant misalignments between their values and their behavior are all too common, even when the consequences do not make headlines. A company

[6]Ann Rhoades, *Built on Values—Creating an Enviable Culture that Outperforms the Competition* (San Francisco: Jossey-Bass, 2011), p. vi.

> may, for instance, claim to honor the value of cooperation and then set up compensation systems that encourage competition. By their actions and decisions, leaders create a culture, and culture always trumps any strategy you try to implement. To inspire top performance, your organization's strategy needs to be well-aligned with values that are meaningful for your customers and employees.

In a development process that is somewhat prescriptive like the Waterfall process, where people are just expected to follow the rules of the process and there is less room for judgment, the implementation of the process may be less sensitive to the culture and values of a company. However, in an agile development process, where people are expected to use a lot more judgment and individual initiative to make decisions and cross-functional synergy is very important, the culture and values of the company take on much more importance and provide:

- An essential framework for guiding people's actions in making decisions;
- A unifying force to build cross-functional teamwork around higher-level objectives that supersede individual organizational goals; and
- A strong motivator for employees engaged in the process who want to see the purpose in what they're doing rather than simply being cogs in the wheel.

There are several possible scenarios:

- Companies that have a well-defined corporate culture and values that work well for their business and are very consistent with an agile development process.

  These companies will naturally have the least difficulty implementing an agile development approach, and the implementation of that approach will likely make the company stronger.
- Companies that have a well-defined corporate culture and values that work well for their business but may not be totally compatible with a pure agile development approach.

  In these companies, it may not make sense to totally change the company's culture to fit with an agile development process, but it does require some adaptation of the agile development process and/or the company's culture to integrate the two.
- Companies that do not have a well-defined culture and values at all and/or have a dysfunctional culture and values that are not really well aligned with their business strategy and/or are not consistently implemented.

  Creating a culture and values for a company where a well-defined culture and values don't exist or changing a company's culture and values that are dysfunctional and not working well can be an extremely difficult thing to do.

Obviously, scenarios #2 and #3 above pose the greatest challenge. In general, the challenge is adapting an agile approach to fit with the company's existing culture and/or adapting the company's culture to fit an agile approach.

In any case, it is important to make an assessment of the company's culture and values to identify any obstacles and inconsistencies that need to be overcome to successfully implement an agile development approach.

Because of the difficulty of changing a company's culture and values, it may make sense to simply implement an agile development process as a first step without working on higher-level integration with the company's business strategy, culture, and values. But it is important to recognize that the agile implementation may have limited success without addressing some of these higher-level integration issues.

Value disciplines

A company's business environment, values, and culture should be determined primarily by the competitive success factors required by the markets it operates in. A mistake some agilists make is that they are so focused on optimizing the development process, they try to adapt the company's culture to fit what the development process requires as if the whole company revolved around the development process. That may or may not be appropriate depending on the business environment that the company operates in.

One of my favorite books on this is *The Discipline of Market Leaders by* Treacy and Wiersema:[7]

> . . . no company can succeed today by trying to be all things to all people. It must instead find the unique value that it alone can deliver to a chosen market . . . One point deserves emphasis: Choosing to pursue a value discipline is a central act that shapes every subsequent plan and decision a company makes, coloring the entire organization, from its competencies to its culture. The choice of value discipline, in effect, defines what a company does and therefore what it is . . .
>
> Three distinct value disciplines have been defined by Treacy: The principle is that companies need to at least be sufficient (not deficient) in all three of them but choose one to excel in as its competitive differentiation.

The three value disciplines that Treacy and Wiersema define are as follows:[8]

1. **Operational Excellence**

   The first of the three value disciplines is *operational excellence*. Companies that focus on operational excellence succeed or fail by offering products and services more efficiently than their competitors can offer them:

[7] Michael Treacy and Fred Wiersema, *Discipline of Market Leaders* (Reading, MA: Addison-Wesley, 1995), p. xiv– xv.
[8] Ibid.

> Companies that pursue this [discipline] are not primarily product or service innovators, nor do they cultivate deep, one-on-one relationships with their customers. Instead, operationally excellent companies provide middle-of-the-market products at the best price with the least inconvenience. Their proposition to customers is simple: low price and hassle-free service. Wal-Mart epitomizes this kind of company, with its no-frills approach to mass-market retailing.

In an operational excellence environment, there is much less room for creativity. At McDonald's, there is only one way to cook the hamburgers, and people are required to follow that process in their day-to-day work. The company does have a defined way of how employees can suggest and implement improvements to improve the process, but creativity in how the hamburgers are cooked is certainly not encouraged. They do the same thing repeatedly at a low cost; that is what operational excellence is, and it is reflected in their culture.

McDonald's culture and values emphasize that people may wear uniforms to work and are heavily trained in doing things the same way all the time. Obviously, there may be more difficulty in implementing an agile development approach in this type of environment. In this type of company, it may make sense for the company to become more agile, but it can't lose sight of its primary value discipline of operational excellence. This is an example of a situation where an agile development approach might need to be adapted to fit the business environment, and it may not make sense to attempt to completely transform the company's business environment to fit with an agile development approach.

2. Product Leadership

The second of the three value disciplines is *product leadership*. Companies that focus on product leadership succeed or fail primarily by innovating products to meet market needs faster and better than their competitors:

> The second value discipline we call product leadership. Its practitioners concentrate on offering products that push performance boundaries. Their proposition to customers is an offer of the best product, period. Moreover, product leaders don't build their positions with just one innovation; they continue to innovate year after year, product cycle after product cycle. Intel, for instance, is a product leader in computer chips. Nike is a leader in athletic footwear. For these and other product leaders, competition is not about price; it's about product performance.

A company whose primary value discipline is *product leadership* needs to create an environment that stimulates creative thought. These companies might be dominated by engineers

who might wear jeans and sweatshirts to work. Obviously, this value discipline is very consistent with an agile development approach.

3. Customer Intimacy

The third of the three value disciplines is *customer intimacy*. Companies that focus on customer intimacy succeed or fail primarily by providing a high level of personalized service to their customers relative to other competitors:

> The third value discipline we have named customer intimacy. Its adherents focus on delivering not what the market wants but what specific customers want. Customer-intimate companies do not pursue one-time transactions; they cultivate relationships. They specialize in satisfying unique needs, which often they, by virtue of their close relationship with—and intimate knowledge of—the customer, recognize. Their proposition to the customer: "We have the best solution for you—and we provide all the support you need to achieve optimum results and/or value from whatever products you buy." Ritz Carlton Hotels is an example of a company that excels at customer intimacy.

In a business environment whose primary focus is on customer intimacy, employees are trained that customer needs and customer service always come first, and their processes need to be flexible to adapt to customer needs. If a customer has a problem, employees are trained to do whatever it takes to solve the customer's problem with or without a process. As an example, I was traveling with my boss some years ago. He wore a shirt requiring cuff links (French cuffs), and he had forgotten his cuff links at home. A maintenance man at the hotel we were staying in quickly went to his shop, found two pairs of nuts and bolts, spray painted them black, and turned them into a set of cuff links. Obviously, you wouldn't find a process anywhere to tell someone how to do that—it requires improvisation and employee initiative, and that's what customer intimacy is all about.

The customer intimacy values of adaptability to meet customer needs are very consistent with an agile development approach; however, you typically find the customer intimacy value combined with other values such as operation excellence in the right proportions.

The idea of value disciplines is that a company can't be *deficient* in any of the three areas, but the company needs to choose one value discipline as the *primary* area of focus to excel in. The value discipline that is chosen as the *primary* competitive differentiator tends to define the whole company and its culture and values.

Many companies have learned to fine-tune their value proposition even further around specific market segments. A very good example of a company that has segmented its customer base is Marriott Hotels. Years ago, there was only one kind of Marriott Hotel. It was the typical high-end luxury hotel that Marriott was famous for at one time. Marriott recognized that its customers had different values:

- Some customers valued that kind of luxury (liked the covers turned down at night with a piece of chocolate placed on the night table by the bed and were willing to pay a premium price for it). The traditional high-end luxury Marriott Hotel fit this value proposition well.

- Other business travelers were more cost-conscious and less concerned about some of these frills. Convenience of getting in and out and staying in a nice comfortable room with the right amenities for business travelers was what was important to them. Marriott Courtyard was developed to fit this need.
- Still other cost-conscious families on a budget wanted a low-cost room and amenities that were designed for a family with children, and Fairfield Inns were created to fill that need.

If Marriott had continued to offer a one-size-fits-all approach, it would have missed the mark in satisfying some of these customers. There would have been a misalignment of their value proposition with what that particular customer segment considered important. This strategy has obviously worked well, as other hotel chains like Hilton have also followed a similar approach.

A misalignment between the primary value discipline a company chooses to pursue and its culture is probably not going to yield optimal results. The same workers in jeans and sweatshirts who were extremely successful in creating new products at a product leadership company are not likely to be as successful cooking hamburgers at McDonald's, where the primary value discipline is operational excellence.

Stephen Covey summed this up very well:

> There is no "right" culture; there is only right fit. Defining the right fit is a process of determining what values are important in your organization's success and committing to them. You must then develop a plan for how people should behave based on those values and put it into practice throughout your organization. The most critical element, of course, is then helping your people adopt those behaviors and live those values, every day.
>
> If you look at the behavior of leaders, you can tell what the real values of a company are. And all too often, those lived values bear almost no resemblance to the stated values. Many leaders want to believe that all they need to do is proclaim a set of values and culture will magically change—but that does nothing to retool the actual values that control day-to-day actions on the front line. Changing those inherent values takes considerably more effort and cannot be accomplished by any leader or set of executives acting alone . . .
>
> If there is one secret shared by companies that create customer-centric cultures, it is that their leaders profoundly understand that people are their biggest asset—and they act on that idea every day. Organizations that value people often don't use it as an advertising slogan. They just do it.[9]

[9]Stephen Covey, quoted in Rhoades, *Built on Values*, pp. ix–x.

He goes on to identify six principles for creating a values-rich culture:

> **Principle #1: You can't force culture. You can only create environment**. A culture is the culmination of the leadership, values, language, people, processes, rules and other conditions, good or bad, present within the organization . . .
>
> **Principle #2: You are on the outside what you are on the inside** . . . you can't force people to smile and treat customers well when they feel ill-treated themselves.
>
> **Principle #3: Success is doing the right things the right way** . . . By defining your values and the behaviors based on them, you also simplify the task of day-to-day decision-making: 'Does this make sense in light of our values?' is all you have to ask.
>
> **Principle #4: People do exactly what they are incented to do** . . . your values will be perceived as hollow and meaningless unless you base compensation and rewards on expressions of behaviors that go along with the values.
>
> **Principle #5: Input = Output. Organizations will only get out of something what they are willing to put into it.** Values maintenance—what we call continuous improvement—is as important as values creation. In other words, you are never really 'done' with culture; you must be always vigilant that no one backslides into old ways.
>
> **Principle #6: The environment you want can be built on shared, strategic values and financial responsibility**. Conscious action, beginning with determining a set of shared values, can set up a necessary condition for encouraging a culture that will make an organization into a leader in its industry . . . Happy talk values that result in spending huge sums of money on questionable programs are not values that are sustainable in the long run. But neither should you let financial concerns derail the process in its infancy.[10]

SUMMARY OF KEY POINTS

Agile development processes are many times implemented as if they were just a development process that is independent of the company's business environment of which they are a part. There are many books on how to implement and optimize an agile development process as a stand-alone process but relatively few books that address the big picture of how to adapt an agile development process to a company's business environment, culture, and values. That is not an easy thing to do because it requires:

[10] Ibid., pp. xvi–xviii.

- A broad-based understanding of business management principles and practices as well as development process principles and practices (both agile and plan-driven)
- A deeper understanding of the principles and practices behind various development approaches (plan-driven as well as agile) to understand how to integrate them in the right proportions to fit a given business environment.

1. **The Impact of Different Business Environments**

   Implementation of an agile development approach creates some new challenges for businesses that they may have never had to face before.

   - If an agile development approach is not well integrated with the company's business environment, culture, and values, it is not likely to be fully effective.
   - Force-fitting a company's business environment and projects to a textbook agile approach is probably not the best approach in many situations. A better approach is to tailor it to fit the company's business environment. This may require blending a combination of principles and practices from a plan-driven management approach, as well as from an agile approach, to fit the situation.

2. **Typical Levels of Management**

   An agile development approach does not necessarily require abandoning some of the traditional plan-driven management approaches such as enterprise-level product/project portfolio management; it just requires some adaptation of the agile development approach to fit with that kind of managed environment or adapting the management environment to fit with agile.

3. **Corporate Culture and Values**

   An agile development process is also very sensitive to the company's culture and values. It can be very difficult to implement an agile development process in a company if there is a misalignment with the company's culture and values or if the company's culture and values are not well defined or are dysfunctional.

DISCUSSION TOPICS

The Impact of Different Business Environments on Agile

1. Why is it more difficult to implement an agile development approach in a company that has a project-oriented business model? How would you go about resolving this difficulty?

Typical Levels of Management

2. What is the potential impact of having a misalignment of different levels of management in an agile implementation?
3. What are the potential solutions?

Corporate Culture and Values

4. Why is corporate culture and values so important to successfully implementing an agile project management approach?

5. Discuss how the concept of value disciplines might apply to some of the companies that you've been associated with. Did the company have a clear idea of what its primary value discipline was? Was it implemented consistently? What were some of the difficulties?

14 Enterprise-Level Agile Transformations

THE WORDS *AGILE TRANSFORMATION* MAY mean different things to different companies. There is also an adoption curve associated with agile, and companies may be at different levels on that adoption curve. An agile transformation may take a long time to implement, depending on the scope and difficulty of the transformation effort.

No matter what the situation, an agile transformation typically involves some level of change, and the transformation should be planned. If the company is just starting out on an agile transformation, it may make sense to pilot the approach on a small scale before attempting to do an enterprise-wide transformation. This chapter provides some guidelines on planning and managing an agile transformation.

PLANNING AN AGILE TRANSFORMATION

The following are things to consider in planning an agile transformation.

Define the goals you want to achieve

I have seen so many companies take a brute-force approach to becoming agile. Doing an agile transformation just for the sake of becoming more agile is not necessarily an appropriate goal. What business value do you expect to get from it?

To many companies, being agile means just going faster. In many cases, that simply means putting pressure on developers to work nights and weekends in Death March projects to get projects done quickly. Agile isn't necessarily only about becoming faster—if it's done correctly, it can have a dramatic positive impact on the way your whole company operates:

- Transforming the nature of the whole business to make it much more dynamic and adaptive to customer needs
- Producing higher quality products that are better aligned with customer needs and provide much higher levels of value to the users
- Dramatically improving the morale and productivity of everyone engaged in the development process because work is done smarter and more efficiently at a planned pace without excessive overtime so that employees are more engaged in the process
- Developing cross-functional synergy among all organizations and functional areas to break down political boundaries to make the whole company more efficient
- Developing products faster

Developing products faster is only *one* benefit of an agile development approach. It's the one that typically gets the most attention, but an agile approach is not necessarily faster. A very important aspect of an agile development process is focusing on business value and being more flexible and adaptive to produce high-value products that are well-aligned with customer needs.

- Sometimes that requires some level of experimentation to iteratively develop a product that is well-aligned with customer needs.
- You also have to be receptive to feedback and changes during the development process. Naturally, that can extend the time required to develop a product; but it can result in a much higher quality product in the end.

Before embarking on any agile transformation, I highly recommend that a company define the goals to be achieved before going too far into an agile transformation. Developing products faster is only one possible goal. Another consideration is this: How agile do you want to be, and how quickly do you want to get there? A pure agile approach will not be appropriate for all companies, depending on the nature of their business. Even if it is an appropriate strategy, it may not be possible to change the culture of the company overnight to adopt a totally agile approach, and a step-wise, incremental approach may make the most sense.

Becoming agile is a journey, not a destination

An agile transformation can take a long time, and there are many trade-offs to be considered. For example, it may not make sense to tailor an agile approach to fit a business environment that you know is dysfunctional. We all know that there are limits on how much (and how quickly) you can change a dysfunctional business environment. Many compromises are often necessary, and the initial implementation may be far less than optimal.

Many times, you have to be realistic about what is achievable, take whatever you can get as a starting point, and continuously improve from there. A focus on ongoing, continuous improvement is

essential—companies that are looking for a one-shot, quick fix are probably going to be disappointed. When I worked in quality management years ago, one of my favorite expressions was quality is a journey, not a destination. The same can be said for helping companies become more agile. It can require a great deal of patience, persistence, and perseverance to implement agile in any business environment.

> In building learning organizations there is no ultimate destination or end state, only a life-long journey.[1]

Using an agile approach to plan and manage the transformation itself as an agile project typically works well:

- Assess the current situation and develop a vision for where the company wants to wind up after the agile transformation. What are the problems and limitations in the current situation? What is the scope of the effort required? How will the company be different when the transformation is complete? It is very important to get consensus on what that vision is, as it can be a unifying force for rallying the company around.
- Develop a product backlog of all the things that might contribute to achieving that vision. Sometimes it can be difficult to get some traction and momentum established. An incremental and iterative approach works well to break up the transformation into bite-sized chunks that enable getting started quickly and show some quick hits to demonstrate progress.
- Prioritize the items in the product backlog and organize the effort into releases and iterations just as you would an agile software development project.
- At each step along the way, engage the customer (senior management) and as many key stakeholders as possible in the effort to get feedback and input.
- Use a continuous improvement approach to assess the effectiveness of the effort and take corrective action as needed to make it as effective as possible.

Develop a culture that is conducive to agile

The cultural transformation required to successfully implement agile in a software development environment is somewhat similar to the cultural transformation that was needed to implement a total quality management (TQM) approach in the automotive manufacturing industry years ago (See Chapter 9 for more details). It can require a major culture shift; however, the principles behind TQM can dramatically improve quality and productivity. Agile has the potential to have a similar but

[1] Peter Senge, *The Fifth Discipline: The Art & Practice of The Learning Organization* (New York: Crown Business, 2006).

even greater impact on a company, but just as with TQM, it may require some rethinking of how the company is managed to create a culture that is more conducive to successfully implementing agile. From a management perspective, many of the problems and challenges that need to be overcome are the same as the problems Peter Senge and Dr. Deming noted that needed to be overcome with TQM. These include:[2]

- Management by measurement (focusing on short-term metrics and devaluing intangibles)
- Managing outcomes (rather than managing the process)
- Overemphasis on predictability and control (to manage is to control)
- Excessive competitiveness and distrust among people and organizations

The one most critical objective to accomplish from an organizational culture perspective is the creation of a Learning Organization environment. This is something that hasn't changed since the implementation of the TQM movement in the 1990s.

> Learning organizations are skilled at five main activities: systematic problem solving, experimentation with new approaches, learning from their own experience and past history, learning from the experiences and best practices of others, and transferring knowledge quickly and efficiently throughout the organization. Each is accompanied by a distinctive mindset, tool kit, and pattern of behavior. Many companies practice these activities to some degree. But few are consistently successful because they rely on happenstance and isolated examples. By creating systems and processes that support these activities and integrating them into the fabric of daily operations, companies can manage their learning more effectively.[3]

This concept is *absolutely essential* to a successful agile implementation. Agile is based on the idea of "fail early, fail often," using experimentation and continuous improvement. This concept also hasn't changed significantly since the 1990s. Here's a quote from Peter Senge in 1996 on this:

> When a group of people collectively recognize that nobody has the answer, it transforms the quality of that organization in a remarkable way. And so we teach executives to live with uncertainty, because no matter how smart or successful you are, a fundamental uncertainty will always be present in your life. That fact creates a philosophic

[2]Ibid.
[3]"Building a Learning Organization," David A. Garvin. Harvard Business Review on Knowledge Management, 1998, p. 52

> communality between people in an organization, which is usually accompanied by an enthusiasm for experimentation. If you are never going to get the answer, all you can do is experiment. When something goes wrong, it's no longer necessary to blame someone for screwing up—mistakes are simply part of the experiment.[4]

Here's another quote on that subject from the same period:

> The big difference between learning and non-learning organizations is not measured in their capacity for error but in their capacity to respond to error. Persistence in error is a sure sign of an organization which, in today's politically correct parlance is "learning-challenged" . . .
>
> But don't be fooled. Dumb organizations are not necessarily staffed by dumb people. In fact, there may be many very smart people in a dumb organization. They are trapped there by a brain-dead organizational apparatus and are often very frustrated. The learning disability usually lies at the organizational level, not with the individual.[5]

Manage change

Change in any organization is inevitable, and an ability to effectively manage change is an essential element of success for any dynamic and growing business. Migration to a more agile approach creates new imperatives for change that will put additional pressure on the need for business transformation.

> The pressures of a truly global economy cause today's business to increasingly rely on their ability to produce software as a key competitive advantage. Whether it's software for managing manufacturing and customer delivery processes or software improving the efficiency of day to day activities, software touches virtually every facet of today's business.
>
> And yet, many CxOs find their software development practices remain little changed from the 1980s. Reliance on prescriptive, plan-based, waterfall-like methods is common despite mountains of evidence that these practices often fail to deliver real value in a timely fashion, and so hamper the company's responsiveness to fast-changing customer requirements and market conditions. And it's not getting easier.[6]

[4]Peter Senge, "Systems Thinking," *Executive Excellence* (January 1996), p. 15.
[5]Barry Sheehy, Hyler Bracey, and Rick Frazier, *Winning the Race for Value*, (New York: American Management Association, 1996), p. 126.
[6]Ken Schwaber, "A Playbook for Adopting the Scrum Method of Achieving Software Agility," www.scrumalliance.org/resource_download/66

The best companies have learned how to reinvent themselves regularly as necessary to remain competitive and have learned that effective management of change is an essential element of success. However, it is not an easy thing to do and is often unsuccessful.

> To date, major change efforts have helped some organizations adapt significantly to shifting conditions, have improved the competitive standing of others, and have positioned a few for a far better future. But in too many situations the improvements have been disappointing and the carnage has been appalling, with wasted resources and burned-out, scared, or frustrated employees. To some degree, the downside of change is inevitable. Whenever human communities are forced to adjust to shifting conditions, pain is ever present. But a significant amount of the waste and anguish we've witnessed in the past decade is avoidable.[7]

It takes courage to make difficult changes. The company needs to embrace honesty and go to those uncomfortable places to identify what needs to be changed. That's the hardest part about any change initiative. If there are sacred cows and things no one wants to discuss or examine, then only symptoms get fixed, not root cause.[8]

Kotter[9] has identified eight errors that companies make in managing change:

1. *Allowing too much complacency*. By far the biggest mistake people make when trying to change organizations is to plunge ahead without establishing a high enough sense of urgency in fellow managers and employees.
2. *Failing to create a sufficiently powerful guiding coalition*. Major change is often said to be impossible unless the head of the organization is an active supporter. What I am talking about here goes far beyond that. In successful transformations, the president, division general manager, or department head plus another five, fifteen, or fifty people with a commitment to improved performance pull together as a team.
3. *Underestimating the power of vision*. Urgency and a strong guiding team are necessary but insufficient conditions for major change. Of the remaining elements that are always found in successful transformations, none is more important than a sensible vision.
4. *Under-communicating the vision by a factor of 10 (or 100 or even 1,000)*. Major change is usually impossible unless most employees are willing to help, often

[7]John Kotter, *Leading Change, With a New Preface by the Author* (New York: Perseus Books Group, 2012; Kindle Edition), pp. 149–151.
[8]Liza Wood, Book Review Comments, personal e-mail, May 28, 2014.
[9]Kotter, pp. 149–151.

to the point of making short-term sacrifices. But people will not make sacrifices, even if they are unhappy with the status quo, unless they think the potential benefits of change are attractive and unless they really believe that a transformation is possible.

5. *Permitting obstacles to block the new vision*. The implementation of any kind of major change requires action from a large number of people. New initiatives fail far too often when employees, even though they embrace a new vision, feel disempowered by huge obstacles in their paths. Occasionally, the roadblocks are only in people's heads, and the challenge is to convince them that no external barriers exist. But in many cases, the blockers are very real.
6. *Failing to create short-term wins*. Real transformation takes time. Complex efforts to change strategies or restructure businesses risk losing momentum if there are no short-term goals to meet and celebrate. Most people won't go on the long march unless they see compelling evidence within six to eighteen months that the journey is producing expected results. Without short-term wins, too many employees give up or actively join the resistance.
7. *Declaring victory too soon*. After a few years of hard work, people can be tempted to declare victory in a major change effort with the first major performance improvement. While celebrating a win is fine, any suggestion that the job is mostly done is generally a terrible mistake. Until changes sink down deeply into the culture, which for an entire company can take three to ten years, new approaches are fragile and subject to regression.
8. *Neglecting to anchor changes firmly in the corporate culture*. In the final analysis, change sticks only when it becomes 'the way we do things around here,' when it seeps into the very bloodstream of the work unit or corporate body. Until new behaviors are rooted in social norms and shared values, they are always subject to degradation as soon as the pressures associated with a change effort are removed.[10]

From my experience, there are generally three key elements that are most critical to any successful change management initiative:

1. *Burning platform* (Not creating a sense of urgency—Kotter's error #1): There's got to be a sufficient level of pain associated with the current situation to convince people that the situation is untenable and must be changed.
2. *Vision for the future* (Underestimating the power of vision—Kotter's error #3): There needs to be a clear vision for what the future will look like when the change is complete.

[10] Ibid.

3. *Progress* (Failing to create short-term wins—Kotter's error #6): There are always naysayers and skeptics who will remain on the sidelines waiting to see if the change is likely to be successful before they jump on the bandwagon. That is why it is important to get started and demonstrate progress as quickly as possible.

My experience has shown that if any of these three elements are not in place and a significant change is needed, it will be very difficult to make any progress. For example, if there is no *burning platform*—if people aren't feeling a sufficient amount of pain from the current situation—the motivation to make a change may be insufficient to do anything differently.

Don't throw the baby out with the bathwater

In migrating to a more agile approach, it isn't necessary to throw away all the wisdom and knowledge gained from a traditional, plan-driven approach. In my first book on agile project management,[11] I used a case study based on Sapient to illustrate this point. Sapient developed a very nice hybrid methodology that combines elements of a plan-driven approach with an agile approach. Sapient recognized that it didn't need to throw away all of its existing traditional processes in order to move to agile.

Erik Gottesman, author of *Growing Agility in a Large and Distributed Enterprise,* explains it this way:

> Lastly, when transitioning to agile methods, one should not believe that every practice in place before the transition is unnecessary or wrong. Your organization must have been doing some things right to begin with; otherwise you wouldn't be in business, would you? We point this out because we have seen a disturbing tendency among organizations to think that in order to be agile, you must unlearn and divorce yourself from all of your old behaviors. This simply isn't true. You must let go of some things. For us, this included evolving our attitude toward fixed-pricing. We also had to get very disciplined about managing WIP and deferring decisions until the last responsible moment in order to avoid speculation and waste.
>
> Conversely, over the course of Sapient's history, we had developed some highly effective processes of our own design for such things as aligning stakeholders around a common vision, eliciting customer requirements, and running Project Management Offices (PMOs). All of this was good and required neither disposal nor significant overhaul in order to integrate with the new techniques introduced.[12]

[11] Charles Cobb, *Making Sense of Agile Project Management—Balancing Control and Agility* (Hoboken, NJ: John Wiley & Sons, 2011).
[12] Erik Gottesman, *Growing Agility in a Large and Distributed Enterprise*, Agile Project Leadership Network, Agile Business Conference, London, UK, 2006, p. 5.

The key message is to use common sense and intelligence when you apply any methodology to your business. It's always a good idea to clearly identify what's working and what's not. Build on what's working and don't fix what's not broken. Too many teams throw everything up in the air, and it's even more painful to put it back together again.[13]

It's ironic that the agile movement started as a revolution against highly prescriptive methodologies like waterfall, and over time some of the agile movement has shifted to the point that agile has become synonymous with particular methodologies like Scrum. There's even a view that if you're not doing Scrum rigidly by the book, you're not agile at all. Scrum is a great methodology (most people call it a framework), but it needs to be applied intelligently in the right context. And, as the usage of agile and Scrum has expanded to enterprise-level agile implementations, Scrum needs to be understood in a much broader context.

Tools can be very important

Some people in the agile community have had somewhat of an aversion to tools and only use paper-based stickies on a wall board to plan an agile effort and track progress. That aversion to using tools probably has its roots in one of the original Agile Manifesto statements:

> Individuals and interactions over processes and tools[14]

However, relying only on paper-based stickies and no other tools has some serious limitations that become significant when you try to scale agile projects to a large enterprise environment.

Communications and Collaboration among Distributed Teams

When teams are distributed, paper-based tools become extremely limited because they are not easily accessible by everyone on the team. Even for teams that are collocated, tools can play a valuable role in coordinating all the efforts of everyone on the team.

Status Tracking and Reporting

In an enterprise environment, projects typically roll up into some kind of overall portfolio management approach that needs to be tracked and managed, and some form of status reporting is essential for that. This capability can also be essential for achieving integration across different projects.

Requirements Management and Traceability

Tools are very valuable for structuring and organizing requirements, development tasks, test cases, and related information in a project. If there is a need to manage traceability of project requirements,

[13] Wood.
[14] Agile Manifesto, http://agilemanifesto.org/.

tasks, and test cases, it would be extremely difficult to do that without some kind of tools that are well suited for performing traceability analysis. If you were using paper-based stickies for user stories, you would have to reenter all of that information into some other kind of tool for analyzing and managing traceability. That would be extremely cumbersome and wasteful. It would also be very difficult to keep those sources of information in synch as the project progressed.

The most important point is that tools should support and serve the project team and enable higher productivity, rather than simply controlling and constraining the actions of the people on the team. Gottesman writes:

> I have often argued that agile methods actually require a higher degree of tooling capability owing to the underlying 'need for speed.' As feedback cycles in an agile/iterative process are so small (a continuous integration build, daily stand-ups, sprints, etc.), you need tooling in order to ensure frictionless flow of information to stakeholders; else, you run the risk of 'gumming up the works' and petrifying the ability to act at the technical/tactical, project/operational, and organizational/strategic levels.[15]

ADAPTIVE PROJECT GOVERNANCE MODEL

At an enterprise level, some form of project governance model may be needed to manage and sustain an agile transformation as well as providing a framework for ongoing management of projects going forward. The words project governance may sound very formal and controlling, but it doesn't need to be that way.

> Governance and agile methods may seem like an oxymoron—but not so. A means to govern is essential for orderly project functioning. Without governance the advantages of adaptive and evolutionary methods could be overwhelmed by functions bolted together haphazardly and rendered operationally ineffective, expensive to maintain, and disadvantageous to customers and stakeholders . . . In its best form governance empowers action and endows decision-making rights to project management and team leaders.[16]

Having an adaptive project governance framework to provide guidance and direction to an agile transformation can accomplish several important objectives:

[15] Gottesman.
[16] John Goodpasture, *Project Management the Agile Way* (Fort Lauderdale, FL: J. Ross Publishing, 2010), pp. 221–223.

- Engage everyone at the right levels on process decisions to build ownership and consensus on the direction of the effort.
- Align the process from top to bottom with the company's business strategy.
- Make the processes real (not a paper exercise) that people really believe in.
- Provide a mechanism for consistency and ongoing process improvement to sustain the transformation.

The problems this model is intended to address are:

- In many agile implementations, there is an assumption that the agile process can be done by the book and there is no need to customize the process to fit the company's business environment. Many times, that is not the case because decisions need to be made to adapt an agile development process to align with the company's business environment. This governance model provides a framework for making those decisions to help achieve buy-in and consensus on the process across the enterprise.
- Agile heavily emphasizes continuous improvement and learning. That learning often takes place within an agile team, and there is no vehicle for capturing lessons learned across project teams and then feeding those lessons learned back in the process for other teams to share.

Figure 14.1 shows that an adaptive project governance model is a general model that can be customized as needed to fit a situation based on the scope of the effort and other factors. The model consists of four levels, but those levels can be customized and/or compressed as necessary.

The following is a description of suggested roles and responsibilities of each of the levels in this model. Of course, these roles and responsibilities should be customized as necessary to fit a given business environment.

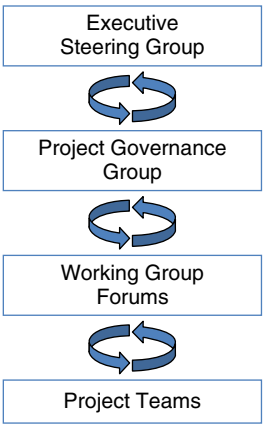


FIGURE 14.1 Multilevel project governance model

Executive steering group

The executive steering group has the following roles and responsibilities in the model:

- Provide high-level strategic direction to align process direction with the company's business strategy.
- Serve as a focal point for building cross-functional integration in the process between all organizations (e.g., IT and the business).
- Make major decisions on project management strategy such as:
  - Determine appropriate balance of agility and control.
  - Set long-term and short-term goals.
  - Lead any necessary change management initiatives across the organization.
  - Periodically review progress of the effort and alignment of processes with the company's business direction.
- Secure the needed buy-in and support from key stakeholders, without whose visible endorsement, the effort may be doomed.

Project governance group

The project governance group has the following roles and responsibilities in the model:

- Approve the selection and design of processes to support the overall project strategy.
- Review and monitor implementation plans, including training, to ensure effective implementation.
- Serve as a focal point for building cross-functional integration in the process within the development teams.
- Monitor the effectiveness of processes and champion ongoing, continuous improvement efforts.
- Review and approve process changes.

Working group forums

The working group forums provide a way for people in similar roles (Scrum Masters, product owners, project managers, business analysts) to share knowledge and best practices, including:

- Provide a forum for reviewing and discussing process effectiveness and leading process improvement initiatives.
- Coordinate activities that span multiple project teams.
- Propose process changes as necessary to the project governance group.

Project teams

The project teams have the following roles and responsibilities in the model:

- Make decisions to tailor and customize the process as needed to fit project requirements within approved guidelines.
- Escalate decisions to deviate from process requirements as needed for further approval.
- Build strong cross-functional teamwork among project teams.
- Manage the effectiveness of the process implementation and identify opportunities for process improvement.

SUMMARY OF KEY POINTS

An agile transformation can mean different things to different companies. It can involve very different levels of scope and complexity, depending on the amount of change that may be necessary. The following is a summary of some key points associated with successfully planning and managing an agile transformation:

1. **Define the Goals You Want to Accomplish**

   Don't make the mistake of force-fitting your business to an off-the-shelf, textbook approach without defining the goals of what you want to get out of it. A better approach is to define the goals you want to achieve and tailor a methodology (or combination of methodologies) to fit those goals. There are also important tradeoffs to consider such as trading off flexibility and adaptability against predictability and control.

2. **Becoming Agile Is a Journey, Not a Destination**

   Developing an agile approach can take a lot of skill, long-term commitment, and determination. Companies that are looking for a quick fix that is easy to implement may be disappointed. It generally works well to use an agile approach during the actual planning and management of the transformation itself:

   - Don't try to do everything at once

> . . . be sensitive to how processes 'spider' across the social graph of the project (across roles, teams, divisions, or even organizations). Just as you would risk-adjust your management of software requirements, you must risk-adjust your transformation effort. Get a few relatively simple 'quick wins under your belt' while also acknowledging and front-loading riskier behavioral changes by starting collaborative cross-functional conversations early. Demonstrate progress and earn the opportunity to press on to more challenging changes.[17]

[17] Gottesman, Erik. Comments on book review

- Develop and prioritize a Product Backlog of goals to accomplish
- Focus on rapid incremental progress to show results quickly

3. **Develop a Culture That Is Conducive to Agile**

   Organizational culture is well known as one of the biggest obstacles to fully implementing an agile transformation. It may or may not be possible to change the culture of a company rapidly (or at all) to adapt to an agile transformation. It is worthwhile to do an assessment of the cultural obstacles that may be necessary to overcome up front in the planning process in order to understand the potential impact on a successful agile transformation.

4. **Manage Change**

   Change management can be a very important aspect of implementing an agile transformation, and it is essential to be sensitive to the factors that are necessary for achieving successful change in an organization at the enterprise level.

5. **Don't Throw the Baby Out with the Bathwater**

   It isn't necessary to throw away all the wisdom and knowledge companies have gained from implementing plan-driven approaches over the years. Some of that knowledge is still worthwhile—it just may need to be implemented in a very different context when you decide to go to agile.

6. **Tools Can Be Very Important**

   One of the original values of agile was Individuals and interactions over processes and tools. For that reason, some organizations try to implement agile using very simple paper-based approach by posting user stories on paper stickies on a white board. That approach works fine for small projects but doesn't scale well for enterprise-level agile projects. Tools can also have a significant impact on team productivity, especially with distributed teams.

7. **Managing an Agile Transformation—Project Governance**

   It is important to put in place some kind of governance model to effectively manage an agile transformation and to provide a forum for ongoing, continuous improvement. The model should provide a framework for engaging all the important stakeholders at the right level in the overall management of the process.

DISCUSSION TOPICS

Planning an Agile Transformation

1. What do you think is the most important factor in developing a successful agile transformation? Why?
2. If you sense that there is some resistance at the enterprise level to implementing an agile transformation, how would you deal with that resistance?

3. How would you go about adapting an agile approach to a company that has an existing management structure built around a traditional PMO and a plan-driven approach to project management?
4. What do you think are the three most critical factors in a change management initiative? Why?

Adaptive Project Governance Model

5. Why would a project governance model be needed at an enterprise level? What functions would it serve?

PART 4

Enterprise-Level Agile Frameworks

PUTTING TOGETHER A COMPLETE top-to-bottom enterprise-level agile solution can be a very challenging task, especially when some of the pieces are not designed to fit together. To simplify the design of an enterprise-level agile implementation, it is useful to have some predefined frameworks that can be modified to fit a given business environment, rather than having to start from scratch to design an overall management approach. Three frameworks are discussed in this section.

Chapter 15 – Scaled Agile Framework

The Scaled Agile Framework developed by Dean Leffingwell provides a fairly complete agile approach for an enterprise-level agile implementation. The Scaled Agile Framework is an excellent approach for companies that want to make a major agile transformation of their overall business.

Chapter 16 – Managed Agile Development Framework

The Managed Agile Development Framework is a hybrid project-level framework that provides a way of doing a more limited implementation of agile in a more plan-driven environment. It provides a hybrid of a plan-driven approach layered on top of an agile development process that can be used to adapt an agile development process to more of a plan-driven organizational approach.

Chapter 17 – Disciplined Agile Delivery Framework

The Disciplined Agile Delivery Framework, a project-level framework developed by Scott Ambler, is in between the above two approaches. It is less adaptive than the Scaled Agile Development framework but more adaptive than the Managed Agile Development Framework.

SUMMARY OF ENTERPRISE-LEVEL FRAMEWORKS

The three enterprise-level frameworks discussed in Chapters 15, 16, and 17 all provide a way of bridging the gap between an agile development process based on Scrum and the higher-level management practices found in many typical businesses. The table below shows a brief comparison of the three frameworks.

Comparison of Enterprise-Level Frameworks

Framework	Pros	Cons
Scaled Agile Framework	■ More of a complete top-to-bottom agile approach	■ Requires a more significant transformation to agile ■ May not be appropriate for many companies
Disciplined Agile Delivery Framework	■ Based on a standard agile development process (Scrum) or lean ■ Provides extensions to the agile development process for scaling to an enterprise level	■ Limited to project-level layer only ■ Not a complete enterprise-level framework but is easily adaptable to existing higher-level management levels
Managed Agile Development Framework	■ Hybrid process with a blend of traditional plan-driven and agile approaches ■ Minimizes the level of organizational transformation needed in organizations with a traditional management structure (Could be done as an interim step in an agile transformation)	■ Limited to project-level layer only ■ Not a complete enterprise-level framework but is easily adaptable to existing higher-level management levels

15 Scaled Agile Framework

THE SCALED AGILE FRAMEWORK (SAFe), developed by Dean Leffingwell, is "an interactive knowledge base for implementing Agile practices at enterprise scale."[1] Figure 15.1 shows a high-level overview of the Scaled Agile Framework.

SAFe is composed of three major levels[2]:

1. *Portfolio Layer*—The Portfolio Layer is the highest and most strategic layer in the Scaled Agile Framework where programs are aligned to the company's business strategy and investment intent.
2. *Program Layer*—The Scaled Agile Framework recognizes the need to align and integrate the efforts of multiple teams that are engaged in large, complex enterprise-level development efforts to create larger value to serve the needs of the enterprise and its stakeholders.
3. *Team Layer*—The Team Layer forms the foundation of the Scaled Agile Framework and is where the fundamental design, build test activities are performed to fulfill the development requirements for each major area of business.

A high-level summary of each of the levels in the Scaled Agile Framework is described in more detail in the following sections. Please refer to the SAFe website for more details on this framework: http://scaledAgileframework.com/

[1] Dean Leffingwell, "Scaled Agile Framework," http://scaledAgileframework.com/.
[2] Ibid.

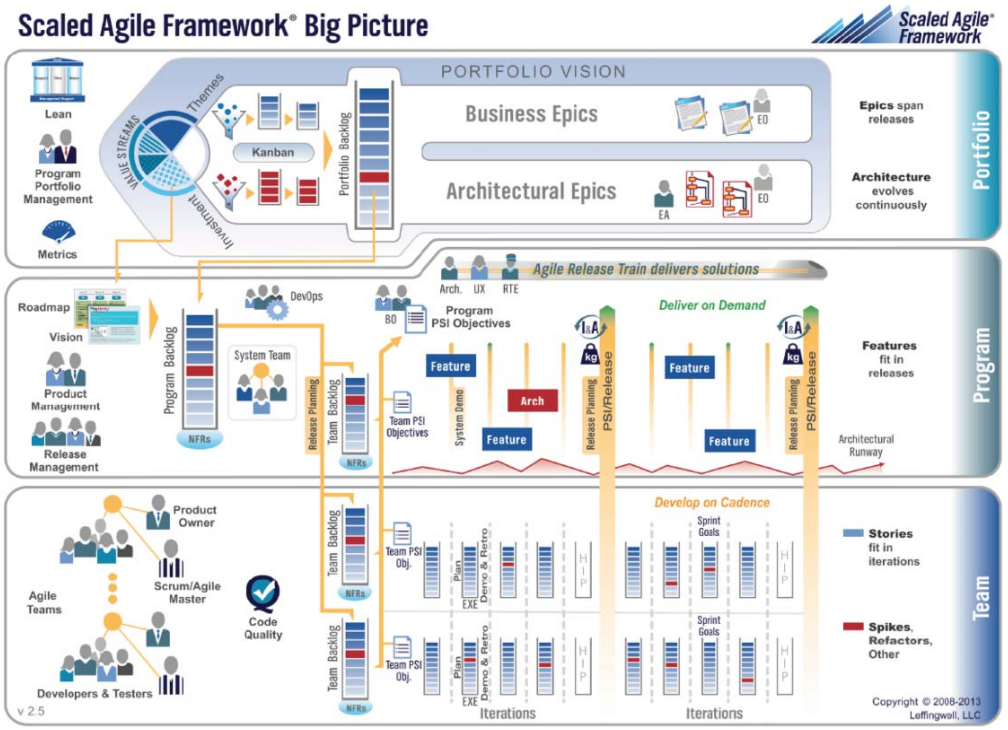


FIGURE 15.1 Scaled Agile Framework

TEAM LEVEL

At a team level, the SAFe uses standard agile principles and practices based on Scrum; however, the SAFe recognizes that implementing an agile development approach at an enterprise level can be much more complex and requires some adaptation:

1. Decisions need to be made about how to best segment and organize the development activities among teams. There are three possible alternatives: feature teams, component teams, or a combination of both.
2. Once those decisions are made, depending on the scope and complexity of the overall program, there is likely to be a need to manage communications and interdependencies across teams to ensure that the activities of all teams are well coordinated and synchronized.
3. Roles can also be more complex. For example, the Scaled Agile Framework recognizes that there may be both a product manager and a product owner involved in a program and distinguishes those roles.

PROGRAM LEVEL

The SAFe recognizes the need to coordinate and integrate the efforts of multiple teams that might be required for large projects and programs, and that is the major purpose of the program level. However, the SAFe does not recognize the traditional metaphor of a project. Instead of a traditional project management structure at the program level, the SAFe uses a more adaptive approach at that level. The major elements of the program level are summarized in Table 15.1.

PORTFOLIO LEVEL

The portfolio level is where the enterprise business strategy gets mapped into investment themes to implement that strategy. The portfolio level employs a number of key constructs, as shown in Table 15.2[3]

[3] "Portfolio Level Abstract," Scaled Agile Framework, updated June 30, 2014, http://scaledAgileframework.com/portfolio-level/.

TABLE 15.1 The SAFe Program Level

Program Level Construct	Description
Vision	The SAFe uses an agile approach for planning the program-level vision. The product management function is responsible for creating the vision, converting that vision into prioritized features in the program backlog, and planning the roadmap of feature delivery. The SAFe recognizes that in a product-oriented company, which is focused on products for external sale, this role would typically be filled by a real product manager. In an internal IT organization, this role would typically be filled by a business analyst. **The inputs for creating the vision consist of:** ■ Investment theme driven strategy ■ Portfolio epics ■ Customer value stream feedback ■ Architectural ■ Team inputs
Roadmap	The SAFe roadmap is similar to a normal agile roadmap in that it provides a plan for allocating product features for releases over a period of time; however, at the enterprise-level it serves the additional function of establishing alignment across all teams. The roadmap uses a rolling-wave planning approach to define its contents.
Release Management	The SAFe recognizes the critical importance of release management at the enterprise level. Releases are generally focused on delivering specific business objectives and features, and a release engineer or program manager provides coordination.

PROGRAM PORTFOLIO MANAGEMENT

SAFe defines a lean and agile approach for the traditional program portfolio management function where the strategic management decisions are made to allocate capacity to investment themes to execute the company's business strategy:

> Program Portfolio Management (PPM) represents the highest-level fiduciary (economics) and content authority (what gets built) in the Framework. These responsibilities typically rest with those business unit/marketing/development executives who have the necessary market knowledge, technology awareness, and best understand the internal financial constraints and external market conditions, and who use that knowledge to drive product and solution strategy. Often, they are assisted by a Project/Program Management Office (PMO), which shares responsibility for program execution and governance.[4]

[4]Ibid.

TABLE 15.2 The SAFe Portfolio Level

Portfolio Level Construct	Description
Investment Themes	Investment themes represent key product or service value propositions that provide marketplace differentiation and competitive advantage.
Epics	Epics are large-scale development initiatives that realize the value of investment themes.
Architectural Runway	In the context of the enterprise's portfolio of products and in the face of a series of shorter, incremental releases, architectural runway is the answer to the big question: What technology initiatives need to be underway now so that we can reliably deliver a new class of features in the next year or so?
Portfolio Vision and Backlog	The portfolio vision provides visibility as to how investment themes will be realized via business epics over time. Epics deliver the value implied by the theme, and they are identified, prioritized, estimated, and maintained in the portfolio backlog.
Kanban Systems	The framework applies two slightly different Kanban systems to manage the portfolio backlogs: one Kanban for business epics, and another Kanban for architectural epics.
Portfolio Management Team	The portfolio management team consists of those individuals who have ultimate responsibility for the lines of business.

SAFe identifies a set of seven transformational patterns that are essential to create the right mindset for effectively implementing a Lean-Agile Program Portfolio Management approach:

SAFe considers the domain of concern for this Program Portfolio Management function to be the set of all programs, both agile and others, below the portfolio level. Within that domain, this function has responsibility for three critical areas:

1. Strategy and Investment Funding
2. Program Management
3. Governance

Each of these three areas is summarized in more detail in Table 15.3.[5]

[5]Ibid.

TABLE 15.3 Lean-Agile Portfolio Management Mindset Shifts

From Traditional Approach	To Lean-Agile Approach
Centralized control	Decentralized decision-making
Project control	Demand management, continuous value flow
Detailed project plans	Lightweight, epic-only business cases
Work Breakdown Structure	Agile estimation and planning
Project-based funding and control	Lean-Agile budgeting and self-managing Agile Release Trains
Waterfall milestones	Objective, fact-based measures and milestones

TABLE 15.4 The SAFe Portfolio Program Management

Program Portfolio Mgt Construct	Description
Strategy and Investment Funding	The purpose of strategy and investment funding is to facilitate implementation of the business strategy through programs that develop and maintain the company's value-added products and services. Value Streams are identified, fostered, produced and continuously improved. Investment funding is allocated to ongoing programs and new initiatives in accordance with business strategy and current Strategic Themes. Additional Lean-Practices help the enterprise meet its economic objectives. **Lean-Agile Budgeting**. Each Agile Release Train has its own budget, which is updated twice annually. By allocating the budget authority to the decision makers on the train – albeit under the auspices of the Business Owners – it is no longer necessary to establish a charter for each new initiative. This avoids overhead and project stop-start discontinuities, the train can make fast and local decisions as needed, within the constraints of the allocated budget. Due to their scope; however, Program Epics still require some level of PPM approval. **Demand Management and Continuous Value Flow**. Overloading any system decreases throughput. If demand isn't managed at the portfolio, the invisible killer of "too much WIP' will limit velocity and quality as teams and individuals thrash from initiative to initiative. Bringing visibility to existing program work and understating the agile program velocities helps manage WIP and insure efficient product development flow. This is managed and supported by implementation of Architecture and Kanban systems, and maintenance and visibility of the Portfolio Backlog. **Epics and Lightweight Business Cases**. In order to provide visibility and economic justification for upcoming cross-cutting work, Business or Architectural Epics are defined and analyzed, each supported by a lightweight business case. Developed by Epic Owners, lightweight business cases provide for reasoning, analysis, and prioritization while avoiding over-specificity.

TABLE 15.4 (*Continued*)

Program Portfolio Mgt Construct	Description
Program Management	Program management supports and guides successful program execution. While this responsibility lies primarily with the Agile Release Trains and the Release Train Engineer, the PPM function can help develop, harvest, and apply successful program execution patterns across the portfolio. In many organizations, the RTE's are part of the PMO, where they can share best practices and common program measures and reporting. In other cases, they report to the development organization. **Self-Managing Agile Release Trains**. Traditional project and program chartering and management activities are replaced by Value Stream based, self-managing and self-organizing Agile Release Trains, each of which provides a continuous flow of value to its stakeholders. **Decentralized, Rolling Wave Planning**. Centralized planning is replaced with decentralized, program and team-based rolling-wave planning via the routine, cadence-based Release Planning activity. **Agile Estimating and Planning**. The formerly too-detailed business cases, too-early requirements specificity and too-detailed work break down structures are replaced with Agile estimating and planning, using the currency of Story points, consistently through the Team, Program and Portfolio.
Governance	Governance functions still exist in agile, otherwise there would be no portfolio-level feedback on investment spend, nor program reporting, nor any means to assuredly communicate and validate important security, regulatory, standards, quality, and release requirements.

16 Managed Agile Development Framework

THE MANAGED AGILE DEVELOPMENT Framework described in this chapter is a project-level framework that is intended to provide a balance of agility combined with some level of predictability and control. It is intended for companies that are unable or not ready to move to a more complete top-to-bottom agile model such as the Scaled Agile Framework. It is a hybrid software development lifecycle model consisting of a blend of an adaptive agile development approach based on Scrum at the micro-level and a more traditional plan-driven approach at the macro-level, as shown in Figure 16.1. It can be easily customized to fit a given project and business environment and can be adapted to companies that have more traditional business and project/portfolio management approaches at a higher level. It generally requires no significant transformation of those higher-level processes.

This approach was developed initially when I was managing a large government project. In order to meet government contractual requirements, we were required to commit to some plan-driven milestones at the program level, and we were required to report earned-value metrics to measure progress against those milestones. On first glance, it may sound impossible to make an agile approach work in that environment, but it worked quite well.

Naturally, there are trade-offs between the level of agility and flexibility to adapt to change at the micro-level and the level of predictability and control at the macro-level. It is important that both the client or business sponsor and the development team agree on those tradeoffs. The framework provides a mechanism for making those tradeoffs by making the macro-level as *thick* or *thin* as you want to fit a given situation.

Instead of having an either-or choice between a fairly rigidly controlled waterfall approach at one extreme and a completely adaptive approach with very limited or no control over costs and schedules at the other extreme, you can customize an approach that provides the desired balance to fit a broad range of situations. The example process flow shown here is intended for large, complex projects

because it is easier to *scale down* a process and to decide to eliminate or minimize activities that are not important to a project than it is to *scale up* and add activities.

MANAGED AGILE DEVELOPMENT OVERVIEW

This framework consists of two layers, as shown in Figure 16.1:

- The macro-level framework is a plan-driven approach, designed to provide a sufficient level of control and predictability for the overall project. It defines the outer envelope (scope and high-level requirements) that the project operates within.
- Within that outer envelope, the micro-level framework utilizes a more flexible and iterative approach based on an agile Scrum approach designed to be adaptive to user needs.

The combination of these two layers is designed to provide a balance of predictability/control and agility:

- The macro-level process provides a high-level framework for achieving some level of predictability and control in the project.
- The micro-level process provides a more flexible and adaptive approach designed to accelerate the development process and optimize the solution to meet user needs.

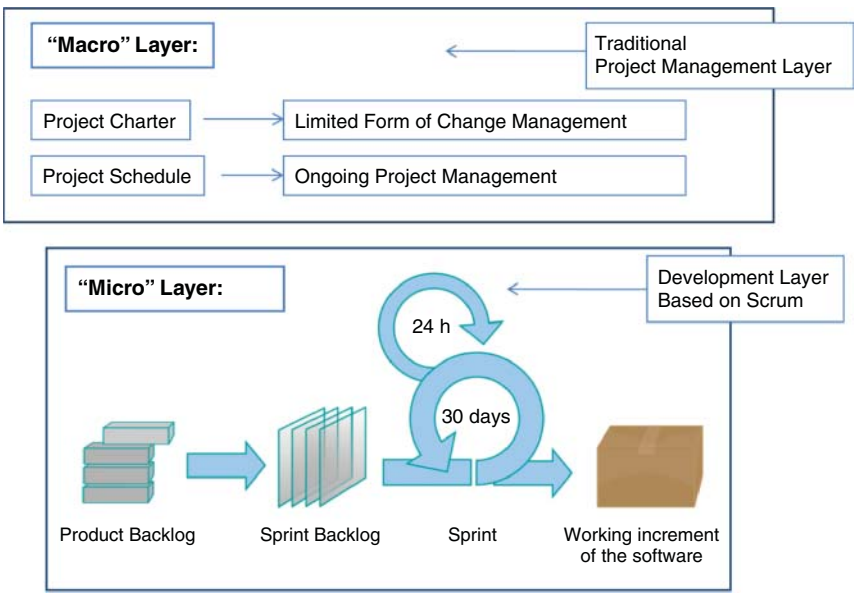


FIGURE 16.1 Framework macro and micro levels

These two different levels will need to be synchronized with each other. This framework is intended to be customized and/or scaled up and down to fit particular projects:

- For small, simple projects, the macro-level can be simplified or eliminated.
- Larger, more complex projects may require more emphasis on a plan-driven approach at the macro-level with more detailed requirements.

Macro-level

At the macro-level, a project charter document is created to define the major deliverables and estimated project milestones based on high-level project objectives. Once that high-level plan is established, it provides a basis for ongoing management of the project. Changes to those macro-level requirements can be controlled as necessary to manage the overall scope of the project.

Micro-level

At the micro-level, requirements are further defined and elaborated in more detail in an iterative approach, with direct participation by the users in an environment that provides more flexibility to adapt to their needs. At a detail level, changes are not finalized until the design has been reviewed and approved by the users at the end of each sprint.

If the work being done at the micro-level results in a change to the higher-level requirements and plan at the macro-level, those changes are fed back to the macro-level process, and the macro-level plan should be adjusted as necessary for the impact of those changes. However, if the process is implemented as it should be, there should be sufficient slack in the macro-level plan to absorb minor changes in requirements at the micro-level so that only significant changes in the micro-level should require replanning at the macro-level.

OBJECTIVES OF MANAGED AGILE DEVELOPMENT

The objectives that this framework is intended to achieve are a combination of the benefits of a traditional plan-driven approach with the benefits of a more flexible and adaptive agile approach.

Plan-driven benefits

1. A well-defined process is consistently implemented and is somewhat predictable.
2. High-level milestones provide a framework for managing user expectations and integration with other activities outside the agile development environment.
3. Scope and cost of projects can be easily and effectively managed.
4. There is improved capacity planning and resource allocation.
5. The process provides a basis for learning and continuous improvement.

Agile benefits

1. *User satisfaction and operational business results*
   - Users are engaged more collaboratively in the effort to define requirements.
   - There is earlier and more direct feedback on design alternatives from iterations and prototyping.
2. *Time-to-market*
   - Requirements are ordered and grouped into releases and iterations.
   - Efforts can be more concurrent and less sequential where possible.
3. *Productivity and efficiency*
   - Team is empowered to tailor the process to fit the project.
   - The process can be optimized for the project.
   - Unnecessary paperwork is reduced
     (more emphasis on direct team involvement and face-to-face communications).
   - Teams are more empowered.
   - The process is a collaborative, cross-functional approach.

Key differences from a typical waterfall approach

There are several aspects of this framework that are significantly different from a typical waterfall approach:

1. *Partnership with the business sponsor and business users*. The typical waterfall model is based on a contractual type of relationship between the business sponsor/users and the development team:
   - Typically with the waterfall model, the business commits to well-defined requirements up front, and the development organization commits to well-defined schedules, costs, and plans to deliver against those requirements. The project manager then takes full responsibility for delivering a solution to meet those agreed-on documented project requirements.
   - In this approach, a high level of understanding of the project scope and high-level requirements exist in the form of user stories at the macro-level. The level of detail in the requirements definition may vary, depending on the nature of the project; however, it is normally not necessary to specify all the details of how a solution will be developed in the project-level planning. Factors that might impact the level of detail that goes into the requirements definition in the project-level planning include:
     - The level of uncertainty in the requirements and the level of confidence needed in the cost and schedule estimates

- The level of risk in the project
- The need for coordination with other groups outside of the agile development environment

  Project-level planning should define the project to a sufficient level to develop an estimate of the project costs and schedule to whatever level of accuracy is required (provided, of course, that the desired level of accuracy is realistic and consistent with the level of uncertainty in the requirements). The project-level planning in the macro layer simply establishes the outer envelope that the project is expected to operate within. It isn't intended to be a rigid contractual agreement. *It is essential to have a level of trust and partnership between the business users and the development team to make that work.*

  Within that envelope established at the macro-level, the business sponsor/users and the project team will jointly own responsibility for further defining the details of the solution, as well as optimizing it to make it successful in the micro-level process. This approach will provide more flexibility to adapt to business requirements as the project progresses and provide a much higher level of assurance that the final solution will meet operational business needs with very high levels of quality and user satisfaction.

2. *Cross-functional team approach*. With a typical waterfall approach, functional managers might primarily manage the efforts of the functional departments that contribute to the project, such as development and QA. Their efforts are typically sequential, and the project manager works to coordinate those efforts into the overall plan.

   - Implementation of this framework requires more of a cross-functional team approach, where the various functional participants in the team (developers, business analysts, testers, etc.) work concurrently as an integrated team with the business users to jointly define, develop, and test the solution throughout the project.
   - The entire project team will also be involved in making collaborative, cross-functional decisions as part of the project-level planning and release-level planning activities, together with the business sponsor, product owner, and business users.

3. *Rolling-wave planning approach*. With a typical waterfall methodology, an attempt is made to develop a detailed plan for the entire project at the front-end of the project. That approach has a number of disadvantages:

   - It delays the startup of the project with an extensive amount of front-end planning.
   - It forces users to *speculate* on what all the requirements for the project should be in detail, very early on in the project, before much information is known about how the solution will be developed.

     The framework defined in this document is based on progressive elaboration and rolling-wave planning. Requirements are generally defined only to a high level in the initial

project planning, and then requirements are progressively elaborated in more detail as the project progresses. The advantages of that approach are:

- It expedites the upfront project planning process.
- It allows the user to defer decisions about detailed requirements until a point where more information is available to make better decisions. In agile terminology, this is called the *last responsible moment*, because it is the latest point possible that a decision can be made without having an adverse impact on the delivery of the solution.
- More direct communication with the users minimizes misunderstanding and miscommunication of requirements, and user requirements can be better integrated with the design effort to optimize the design.
- It minimizes the effort required to document requirements along with translation errors that might result from misunderstanding documented requirements. (Note that this approach does not eliminate the need for documented requirements.)

4. *Iterative approach*. With a typical waterfall methodology, the entire solution is usually designed, developed, and tested sequentially. The disadvantages of that approach are:

   - Slow overall development time due to the sequential nature of all activities.
   - Feedback on problems and defects may not be discovered immediately.
   - Opportunities to get early user feedback and inputs may be limited because users are not involved at all in the development effort to provide feedback, and user acceptance doesn't occur until the very end of the project.

The advantages are:

- Using a more iterative approach will provide earlier realization of business benefits and greater ROI.
- Using a more iterative approach will provide faster overall development time because more activities can be overlapped and concurrent.
- Defects and problems can be detected and corrected much earlier.
- The user is much more directly engaged in the project as it progresses. This provides direct and immediate feedback and will result in a higher level of assurance that the project will meet user needs.
- Some efforts can be performed concurrently rather than sequentially, based on a risk assessment by the project team. For example, the planning phase for each release can typically be overlapped with the execution phase of the previous release.

FRAMEWORK DESCRIPTION

Project organization and work streams

This approach is dependent on using relatively small teams of approximately 8 to 10 people each to perform the design, development, and testing of the solution. For larger projects, the overall project

can be broken up into work streams, and each work stream can be assigned to a team with a team leader/Scrum Master. Each team will consist of the minimum core team required to design, develop, and test the requirements associated with that particular work stream (developers, testers, and business analysts). Specialized resources such as data architects, database developers, and system architects can be centralized as needed to provide support to all work stream teams.

High-level process overview

Each project or work stream will normally be broken into releases and iterations, and each release will typically result in a deliverable product to production as depicted in Figure 16.2. The purpose of breaking the project into releases is to accelerate the delivery of critical features. The features and user stories to be included in each release should be prioritized to deliver the functionality that provides the highest level of value as early as possible. That allows critical functionality to be delivered quickly, without waiting for 100 percent of the total functionality required for the ultimate implementation of the system. An important assumption in this is that the deliverables within each release are sufficiently independent of each other that they can be delivered to production in increments.

Project-Level Planning

Project-level planning will normally consist of:

- Breaking up the project into work streams if necessary for large projects. Erik Gottesman puts it this way:

> From a purely software engineering perspective, it's generally good practice to structure the work-streams/work cells along similar lines as the module structure or decomposition of the software product's logical architecture, as that structure should evidence SOLID principles (strong cohesion, single responsibility, loosely coupled, etc.). By adopting this approach, you guarantee a level of independence between the work-streams that empowers them to act more independently and build strong 'contracts' between one another. The alternative is a disastrous scenario wherein the individual teams achieve zero velocity and have no product to show for themselves at the end of each iteration, resulting from poorly defined product boundaries and unrealized dependencies.[1]

- Organizing the project team and kicking off the project.
- Defining a project charter that includes the business objectives to be accomplished, risks, assumptions, and dependencies, and key milestones to be accomplished. See Project Charter Document Template
- Defining and ordering the product backlog required for the project or work stream and developing a high-level estimate of the effort required for each product backlog item in terms of story points.

[1] Erik Gottesman, Comments on book review

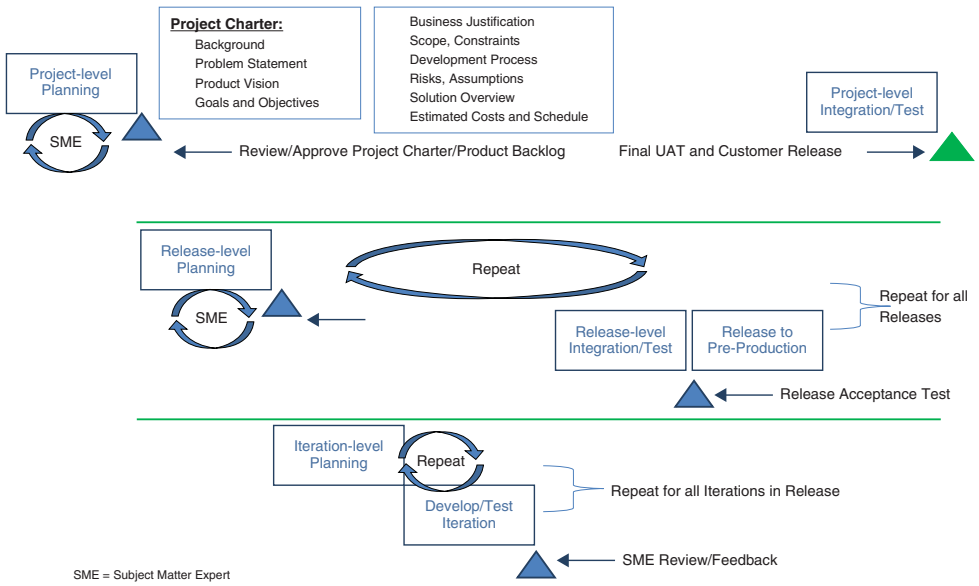


FIGURE 16.2 High-level framework overview

- Tentatively allocating the user stories in the product backlog to releases if necessary.
- Developing a high-level plan with resource requirements for completing all of the requirements of that work stream, including identifying any dependencies on shared resources outside of the project team.
- Identifying and resolving any significant issues and uncertainties that must be resolved prior to starting the project and mitigating any significant risks.

Project-level planning will normally be performed once at the beginning of the project, and that plan will be updated as the project progresses. The project-level planning process is shown in Figure 16.3.

Release-Level Planning

Release-level planning will normally consist of allocating the user stories in each release to iterations, estimating the schedule required to complete each of the iterations required for that release, and resolving any major questions or issues that must be resolved prior to beginning the effort required for that release. The results of the release-level planning in each project or work stream will normally be fed back into the macro-level process to ensure that the micro-level project-level planning is consistent with macro-level project goals and deliverables. The release-level planning process is shown in Figure 16.4.

Iteration-Level Process

Each release may be further broken down into iterations. (If a project contains only releases with no iterations, the release-level planning and iteration-level planning will be combined.) An iteration is a portion of a release that is segmented from the rest of the release for the purposes of optimizing the delivery of the overall release. An iteration normally produces some deliverable functionality that can be demonstrated to the user to show progress and also to get user feedback and inputs; however, an iteration may not be releasable to production. The major benefit of segmenting a release into iterations is to provide a mechanism for getting early user feedback and inputs on the results of each iteration.

Fixed-length iterations are preferred because they allow the team to establish a cadence, but iterations may or may not be fixed-length time-boxes. They may be sized to fit the level of effort required to complete the user stories that the team included in that iteration. The length of iterations should be kept relatively short to demonstrate progress and get user feedback quickly. The project team will be responsible for planning and performing each iteration. Iteration-level planning will typically consist of:

- Clearly identifying all user stories to be included in that iteration.
- Resolving any major questions that need to be resolved prior to starting the iteration and identifying the level of user input required for completing each the task during the iteration. If there are significant uncertainties that cannot be easily resolved prior to the start of the iteration, a special

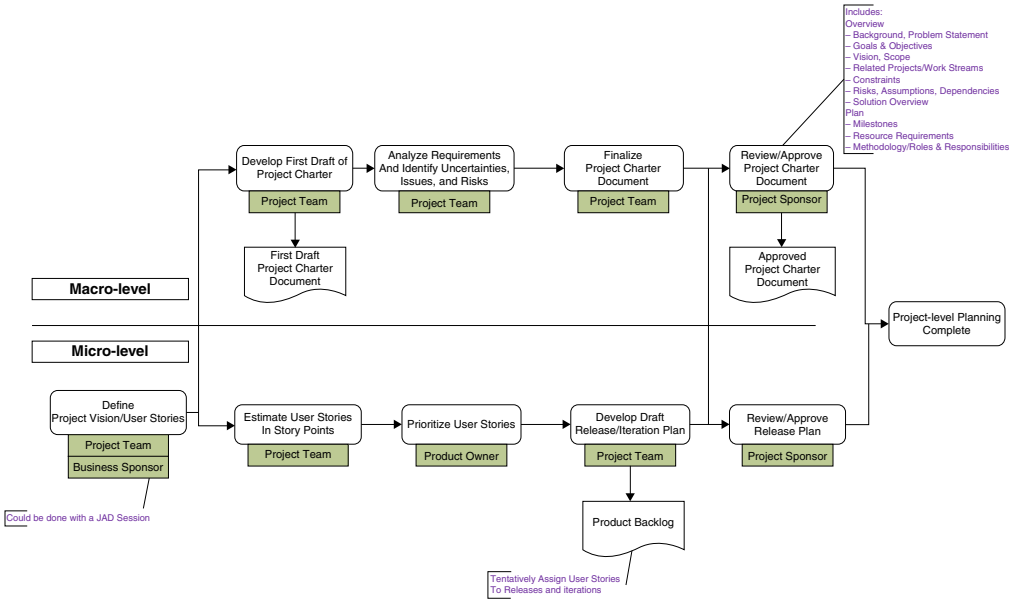


FIGURE 16.3 Project-level planning

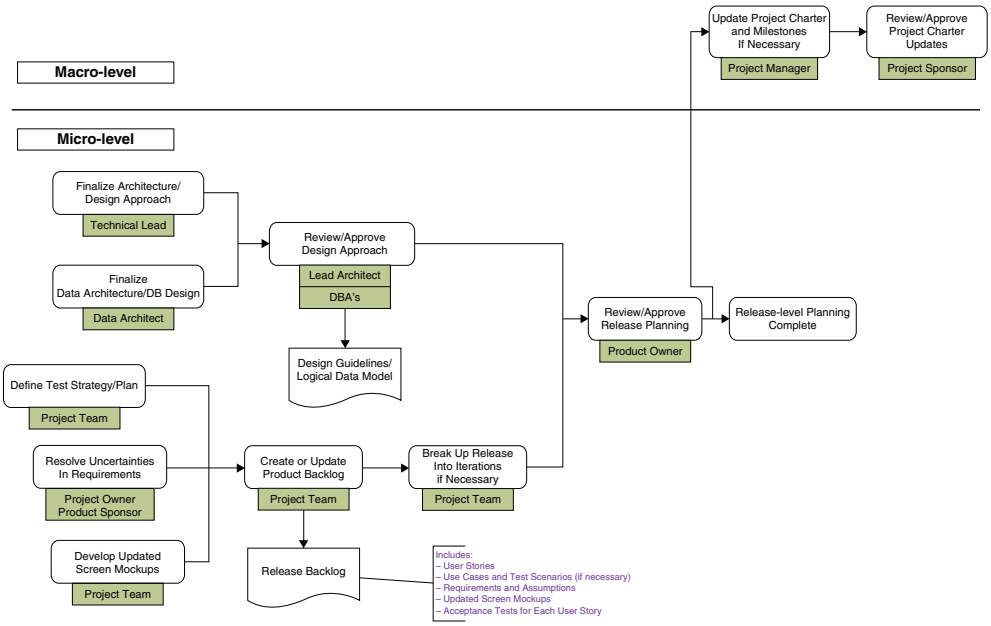


FIGURE 16.4 Release-level planning

- iteration or spike should be planned to resolve those uncertainties prior to beginning the normal development iteration.
- Defining user acceptance criteria for each user story to be included in the iteration.
- Identifying the development tasks required for completing that iteration including testing and allocating the tasks to individual developers.
- Estimating the time required for completing each development task in the iteration in hours. The estimate must include all activities required for development and testing of the task (see definition of "done").

The results of each iteration will normally be as fully tested as possible and accepted by the user prior to completion of the iteration:

- Developers will be responsible for designing, developing, and unit-testing as the software is developed.
- QA testers will be responsible for any system testing to verify that the results of the iteration meet requirements.
- The users will be asked to perform limited user acceptance testing at the completion of the iteration to validate that the results of the iteration meet user needs.

The iteration-level planning process is shown in Figure 16.5.

Requirements management approach

The users will be heavily and directly engaged in the development effort as the project progresses to provide direct feedback and inputs. In cases that require a significant amount of user input, a prototyping approach may be used to further define and elaborate user inputs as the design effort progresses. An example of a situation that would require a significant amount of user input might be the design of a graphical user interface (GUI).

A progressive elaboration approach will be used to define requirements. At the beginning of the project, a planning session such as a *joint application development* (JAD) session is normally held jointly with the business users, the project team, and all major stakeholders. The result of that session is typically a vision for the solution and a list of high-level features for the solution, usually in user story format. That feature list will become the product backlog, which will be used to plan and drive the development effort. The product backlog will be ordered and broken down into releases and iterations, primarily based on the relative importance of the items in the backlog to the users. However, other factors such as the stability of the requirements associated with the items might be important considerations. (For example, some items that are known to have unstable requirements may be deferred to allow more time for the requirements to stabilize.)

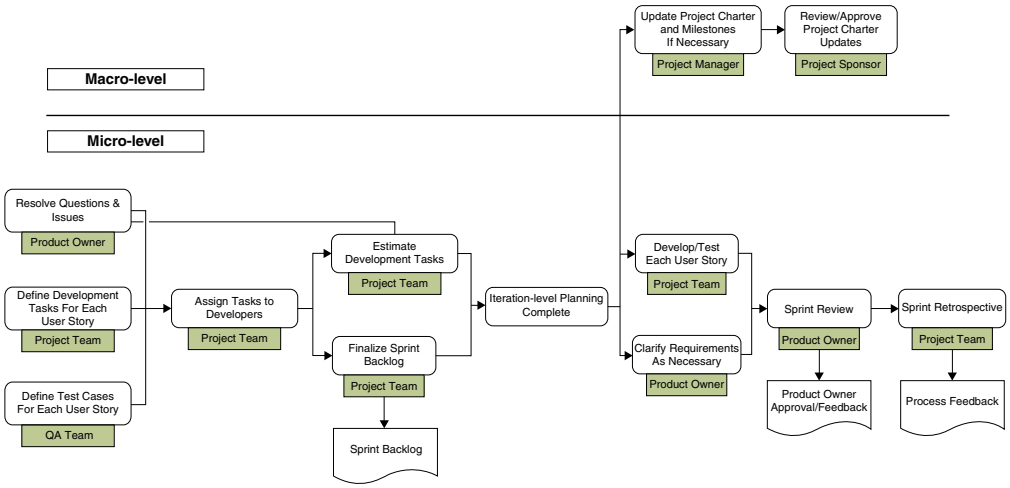


FIGURE 16.5 Iteration-level planning

The requirements definition effort for the project will normally be organized as follows:

1. *Project-level*—During the project-level planning of the project, the high-level requirements for the entire project will be identified to the extent necessary to define the requirements at a feature or user-story level. Those requirements will be tentatively assigned to releases.
2. *Release-level*—The release-level planning for each release will primarily focus on:
   - Resolving any outstanding questions and issues associated with the *release backlog* to the extent necessary to start design and development of each iteration in the release. The release backlog is the subset of product backlog items included in the current release.
   - Defining the requirements for subsequent releases only to the extent that they might impact the implementation of the current release
   - For example, during the development of the detailed requirements for release 1, the detailed requirements for releases beyond release 1 will only be defined to the extent that they might impact release 1. As an example, *hooks* might need to be put in the architecture required for release 1 to accommodate features that are expected later in releases 2 and 3.
3. *Iteration-level*—At an iteration level, the detailed requirements for each iteration will be elaborated only to the extent needed to complete the design and development effort for that iteration. Depending on the functionality included in that iteration, the requirements elaboration might be completed prior to the design and development effort, or it might be completed concurrently with the actual design and development effort. *It is important to note that requirements elaboration means clarification of details of how a requirement should be implemented. Major changes to requirements should not be allowed once an iteration is in progress*.

It is assumed that detailed requirements will continue to evolve as the project progresses without formal change control; therefore, formal change management will normally not be done at the micro-level, except for changes that impact the high-level requirements. The business user representative (product owner) can approve at the micro-level any changes to the detailed project requirements that do not significantly impact the high-level project plan. Any change that significantly impacts the high-level scope and direction of the project should be approved by the project sponsor.

Project scheduling approach

An implication of breaking the project up into releases and iterations using a rolling-wave planning approach is that, at the onset of the project, only an estimate of the overall project schedule will be known. For example, at the beginning of release 1, an estimate of the schedule for completing the release 1 requirements will be made, but normally only a rough estimate of the schedule for subsequent releases and iterations will be available. As each release and iteration is completed, the schedule for subsequent releases and the overall project schedule will be progressively defined and updated.

At the macro-level, an overall project schedule will be developed and maintained throughout the project, but it is understood that because the requirements will be progressively elaborated only at the micro-level within each release and iteration, those estimates of the project schedule are likely to change. Adjustments to the scope of what is included in each release and iteration may be needed to synchronize the deliverables with the macro-level project schedule.

Project management approach

The following is a description of the project management approach:

1. The high-level project charter document defined at the macro-level will typically define the overall scope and vision for the project and will also include milestones for measuring progress. This high-level project charter document defines the *envelope* that the project is expected to operate in; however, it is understood that the scope may change as the project progresses.
2. The high-level project charter document will normally be prepared by the project team, with close collaboration with the business and approved by the project sponsor. Once approved, any significant deviations from these high-level artifacts must be approved by the project sponsor.
3. At the micro-level, a business user representative (product owner) and the project team will jointly take responsibility for the execution of the project as long as the project stays within the envelope defined by the high-level project charter document. The business user representative (product owner) must be a knowledgeable subject matter expert in the area being developed and must be empowered by the project sponsor to make decisions on how the detailed requirements associated with how the functionality will be implemented. This might be implemented in different ways:
   - In the simplest case, the project sponsor, product owner, and business user representative could be the same person.
   - In a more complex case, you might have a project sponsor who has ultimate approval authority for the project and a product owner who has decision-making authority on the details of how the project is implemented.
   - You might or might not have business user representatives in addition to the product owner to represent different stakeholder needs. (The product owner may serve as representing all business needs.)
4. The project team will normally track progress against the high-level milestones at the macro-level and periodically publish status reports to the project sponsor.

   At the macro-level, the major milestones to be tracked might include completion of:

   - Project-level planning
   - Release-level planning for each release
   - Completion of each iteration within the release

- Limited UAT for the deliverables for each iteration
- UAT for the deliverables for each release

At the micro-level, the activities to be tracked might include:

- Sprint burn-down charts to monitor completion of development tasks
- Release burn-down charts to monitor completion of development and testing for all project requirements
- Test plan progress to monitor progress of completing test cases required by each test plan
- Resolution of any bugs

Communications approach

Most of the communications within the project team and with business user representatives will heavily rely on direct communications (either face-to-face or phone conferences). Table 16.1 is a suggested format that can be used for managing project communications.

TABLE 16.1 Format for Managing Project Communications

Meeting	Description	Frequency
Daily Scrums	A daily Scrum is an agile practice and is used at the team level within each work stream. The team leader (Scrum Master) responsible for each work stream will facilitate these meetings. ■ In a pure agile environment, the agenda for this meeting is very simple and follows standard Scrum guidelines. Everyone on the team answers three questions: ■ What did you accomplish yesterday for the project? ■ What are you planning to work on today? ■ What obstacles are in your way? ■ In some cases, such as when an offshore development team is involved, there may be a higher need for communications, and going beyond these basic questions may be necessary. ■ To keep these meetings short and focused, the rule should normally be not to attempt to resolve issues and questions in this meeting unless they are very simple things to resolve. If anything comes up that requires a significant amount of discussion, another meeting should be scheduled to discuss it outside of the Scrum.	Daily
Other Project Meetings	Other meetings will be scheduled as necessary to resolve specific issues that cannot be resolved in the Daily Scrum meetings.	As needed
Weekly Status Meeting	This will typically be a weekly meeting to review project status with the project sponsor. This meeting is primarily focused on reviewing macro-level progress with the project sponsor, where the other meetings are more at the micro-level. For small projects, this meeting might not be necessary.	Weekly

In large projects requiring multiple work streams, there will naturally be a need for additional weekly meetings to coordinate the efforts of multiple teams.

ROLES AND RESPONSIBILITIES

Table 16.2 is a suggested description of the major roles and responsibilities of the key participants in this process. Note that some of these roles may be combined in actual practice (e.g., the Scrum Master and project manager may be the same person, and the product owner and the business analyst may be the same person). These roles and responsibilities are intended to be customized as necessary for a given business and project environment.

TABLE 16.2 Roles and Responsibilities of Key Participants

Role	Responsibility
Business Sponsor	The business sponsor has ultimate responsibility for the success of the project or program from a business perspective, including: ■ Providing direction on business objectives that the project or program must achieve to maximize the benefits of the project or program to the business ■ Ensuring that the appropriate business personnel are fully engaged in defining and prioritizing requirements for deliverables ■ Reviewing and approving all deliverables prior to implementation ■ Resolution of any issues that cannot be resolved by the project team
Business Process Owner(s)	Business process owner(s) are responsible for the execution of the business processes that are affected by the project. Often, they will be responsible for owning and running the software solution. Business process owners are responsible for: ■ Planning and implementing any business process changes that may be required by the solution ■ Ensuring that the solution is consistent with new business processes ■ Planning the cutover of the solution to ensure that any business process changes and other important requirements outside of the development effort, such as user training, are synchronized with the implementation of the solution
Subject Matter Expert (SME)	Subject matter experts provide domain-specific knowledge in an area that is relevant to the project, including: ■ Representing user needs ■ Clarifying project requirements as necessary ■ Reviewing and signoff on business-related project documents

(*continued*)

TABLE 16.2 (*Continued*)

Role	Responsibility
Stakeholder	A stakeholder is a person, group, organization, or system that affects or can be affected by an organization's actions. A stakeholder is identified as someone who has a direct or indirect interest in a project. Stakeholders can range from the project sponsor to an end user, and a stakeholder can be any person who can affect or be affected by the products of a project, either during the project or after the project has been completed. Stakeholders are responsible for: ■ Representing their area of interest and providing inputs to the project requirements and project planning effort to ensure that there is no unexpected impact to their area of interest ■ Ensuring that the solution is consistent with the requirements in their area of interest and validating that the solution has no unexpected impact during the testing and implementation of the solution
Project Manager	The project manager is responsible for: ■ Leading the development of an overall project charter and project plan for the project ■ Integrating the activities within the scope of the project into the overall project plan and ensuring that they are well aligned with the project's business objectives ■ Tracking and reporting the status of all activities within the scope of the project ■ Resolving issues or escalating any issues that cannot be resolved within the project manager's responsibility ■ Taking full responsibility for the management of their assigned project(s) in accordance with any relevant processes, including: ■ Monitoring and guiding each project through to completion, using specific techniques and procedures to establish the framework and structure of the project ■ Keeping project stakeholders informed and involved in project decisions ■ Anticipating changes required in project plans and processes and recommending alternative approaches if necessary ■ Considering impacts to other projects and coordinating the planning and implementation of their project(s) with other project teams via project managers as necessary ■ Taking the initiative to achieve value-added results, within scope of the PM responsibility ■ Leading the project team in taking accountability for work products and ensuring quality and timely delivery of end results

TABLE 16.2 (*Continued*)

Role	Responsibility
Scrum Master	The Scrum Master: ■ Serves the team, not the reverse ■ Responsible for maintaining Scrum values and practices ■ Facilitates most meetings ■ Removes impediments ■ Tracks metrics (i.e., burn-down chart) ■ Communicates with management (status, impediments) ■ Shields the team from external interferences
Product Owner	The product owner: ■ Creates and maintains the product backlog ■ Prioritizes and sequences the backlog according to business value or ROI ■ Assists with the elaboration of epics, themes, and features into user stories that are granular enough to be achieved in a single sprint ■ Conveys the vision and goals at the beginning of every release and sprint ■ Represents the customer; interfaces and engages with all customer stakeholders ■ Participates in the daily Scrums, sprint planning meetings, and sprint reviews and retrospectives ■ Inspects the product progress at the end of every sprint and has complete authority to accept or reject work done ■ Can reorder and redefine the product backlog at the end of every sprint to reflect additions and changes that do not impact the macro-level scope of the project. (Significant changes that impact the macro-level scope may need to be approved by the project sponsor.) ■ Terminates a sprint if it is determined that a drastic change in direction is required
Business Analyst	The business analyst (if required) is the primary resource for the day-to-day efforts of eliciting, analyzing, documenting, and validating the business requirements. The business analyst: ■ Analyzes and scopes the project solution and works with project managers, product owner, and business sponsors to clarify the level and complexity of the business analysis effort needed for the project. (Note that on some projects, the business analyst may also play the role of the product owner.) ■ Selects the appropriate elicitation technique to efficiently identify critical requirements and asks the right questions through the use of interviewing techniques developed specifically for business analysis elicitation

(*continued*)

TABLE 16.2 (*Continued*)

Role	Responsibility
	■ Plans an approach for analyzing, categorizing, and managing requirements; determines the level of formality required and considers options for documenting and packaging requirements based on project type, priorities, and risks ■ Analyzes and refines business and functional requirements from a business perspective, identifying any issues and questions that must be resolved and verifying that requirements are testable ■ Builds strong relationships with project stakeholders and conducts effective requirements reviews to improve the quality of requirements deliverables ■ Anticipates issues, thinking proactively and using critical-thinking skills to plan stakeholder elicitation sessions The business analyst may also perform the role of a business systems analyst: ■ Analyzes and refines business and functional requirements from a systems perspective, identifying any issues and questions that must be resolved and verifying that requirements are testable

17 Disciplined Agile Delivery Framework

THE DISCIPLINED AGILE DELIVERY (DAD) Framework is a hybrid agile approach developed by Scott Ambler:

> Many organizations start their agile journey by adopting Scrum because it describes a good strategy for leading agile software teams. However, Scrum is only part of what is required to deliver sophisticated solutions to your stakeholders. Invariably teams need to look to other methods to fill in the process gaps that Scrum purposely ignores . . . "
>
> DAD is a hybrid approach which extends Scrum with proven strategies from Agile Modeling (AM), Extreme Programming (XP), Unified Process (UP), Kanban, Lean Software Development, Outside In Development (OID) and several other methods. DAD is a non-proprietary, freely available framework. DAD extends the construction-focused lifecycle of Scrum to address the full, end-to-end delivery lifecycle from project initiation all the way to delivering the solution to its end users.[1]

- The Disciplined Agile Delivery Framework developed by Scott Ambler is designed around a number of important characteristics:[2]
- People first
- Learning-oriented
- Agile
- Hybrid
- IT solution-focused

[1]Blog, "Disciplined Agile Delivery—Introduction to DAD," posted by Scott Ambler, 2014, http://disciplinedagile delivery.wordpress.com/introduction-to-dad/.
[2]Scott W. Ambler and Mark Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise* (IBM Press) (Upper Saddle River, NJ: Pearson Education, 2012; Kindle Edition), pp. 533–535.

TABLE 17.1 Goals for DAD Phases

Inception Phase	Construction Phase	Transition Phase
■ Form initial team. ■ Identify the vision for the project. ■ Bring stakeholders to agreement around the vision. ■ Align with enterprise direction. ■ Identify initial technical strategy, initial requirements, and initial release plan. ■ Set up the work environment. ■ Secure funding. ■ Identify risks.	■ Produce a potentially consumable solution. ■ Address changing stakeholder needs. ■ Move closer to deployable release. ■ Maintain or improve upon existing levels of quality. ■ Prove architecture early.	■ Ensure the solution is production ready. ■ Ensure the stakeholders are prepared to receive the solution. ■ Deploy the solution into production.

- Goal-driven
- Delivery-focused
- Enterprise aware
- Risk- and value-driven
- Scalable

DAD defines a lifecycle model consisting of three distinct phases. The goals for each phase in the lifecycle model are described in Table 17.1.[3]

The overall Disciplined Agile Delivery lifecycle model is show in Figure 17.1.

The DAD Framework provides a better foundation for scaling agile in a number of ways. It is described in the "Introduction to DAD," quoted here:

> - First, it promotes a risk-value lifecycle the riskier work early in an endeavor in order to help eliminate some or all of the risk, thereby increasing chance of project success. Some people like to refer to this as an aspect of "failing fast" although we like to put it in terms of succeeding early."
> - Second, DAD promotes self-organization enhanced with effective governance based on the observation that agile project teams work within the scope and constraints of

[3] Ibid.

a larger, organizational ecosystem. As a result DAD recommends that you adopt an effective governance strategy that guides and enables agile teams."

- Third, DAD promotes the delivery of consumable solutions over just the construction of working software. In addition to producing software DAD teams also create supporting documentation, they need to upgrade and/or redeploy the hardware the software runs on, they potentially change the business process around the usage of the system, and may even affect changes to the organization structure of the people using the system."
- Fourth, as described earlier DAD promotes enterprise awareness over team awareness."
- Fifth, DAD is context-sensitive and goal driven, not prescriptive. One process size does not fit all, and effective teams tailor their strategy to reflect the situation they find themselves in.[4]

An important differentiator of the Disciplined Agile Delivery Framework is that it explicitly recognizes the following:

> [A]gile teams, just like other types of teams, are governed. Governance establishes chains of responsibility, authority, communication, and funding in support of the overall enterprise's goals and strategy. It also establishes measurements, policies, standards and control mechanisms to enable people to carry out their roles and responsibilities effectively. You do this by balancing risk versus return on investment (ROI), setting in place effective processes and practices, defining the direction and goals for the department, and defining the roles that people play with and within the department.[5]

Another important differentiator of the Disciplined Agile Delivery Framework is a recognition of the need to be *enterprise aware* and go beyond Scrum to adapt the Scrum development process to an enterprise level. Ambler has defined enterprise awareness as follows:

> People are motivated to consider the overall needs of their organization, to ensure that what they're doing contributes positively to the goals of the organization and not just to the suboptimal goals of their team. This is an example of the lean principle of optimizing the whole, in this case the organization, over local optimization within just the team.[6]

[4] "Introduction to DAD."
[5] Scott Ambler, "Going Beyond Scrum: Disciplined Agile Delivery," white paper series, ©2013 Disciplined Agile Consortium, http://disciplinedagileconsortium.org/Resources/Documents/BeyondScrum.pdf, p. 6.
[6] Ibid, p. 10.

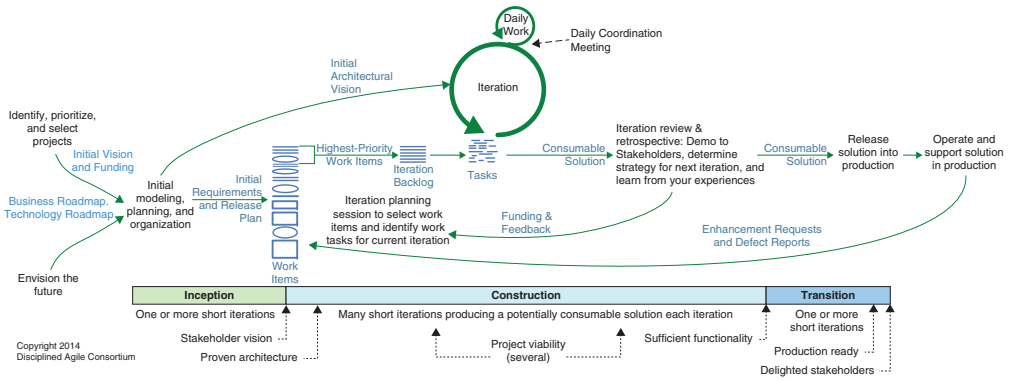


FIGURE 17.1 Disciplined Agile Delivery lifecycle model

Ambler has summed up the most important characteristics of enterprise awareness as follows:[7]

1. **Work closely with enterprise professionals.** It takes discipline to work with enterprise professionals such as enterprise architects, data administrators, portfolio managers, or IT governance people who may not be completely agile yet, and have the patience to help them. It takes discipline to work with your operations and support staff in a DevOps manner throughout the lifecycle, particularly when they may not be motivated to do so.
2. **Adopt and follow enterprise guidance.** Your organization may have, or hopes to one day have, a range of standards and guidelines (guidance) that it wants delivery teams to adopt and follow. This may include guidance for coding, user interface development, security, and data conventions to name a few. Following common guidance increases the consistency and maintainability of your solutions, and thus your overall quality.
3. **Leverage enterprise assets.** There may be many enterprise assets, such as reusable code, patterns, templates, and data sources that you can use and evolve.
4. **Enhance your organizational ecosystem.** The solution being delivered by a DAD team should minimally fit into the existing organizational ecosystem—the business processes and systems supporting them – it should better yet enhance that ecosystem. Furthermore, experienced DAD teams will even fix problems that they run into via proven refactoring techniques, thereby reducing the costs of maintaining these assets and extending their useful lives.
5. **Adopt a DevOps Culture.** DAD teams will work with operations and support staff closely throughout the lifecycle, particularly the closer you get to releasing into production. This collaboration reduces the risk of deployments and ensures a smooth transition to support groups. DevOps philosophies and strategies are baked right into DAD.
6. **Share learnings.** DAD teams are learning oriented, and one way to learn is to hear about the experiences of others. The implication is that DAD teams must also be prepared to share their own learnings with other teams. To do this, organizations might choose to support agile discussion forums, informal presentations,

[7] Ibid, pp. 10–12.

training sessions delivered by senior team members, and internal conferences to name a few strategies.

7. **Adopt appropriate governance strategies.** Effective governance strategies should enhance that which is being governed. An appropriate approach to governing agile delivery projects, and we suspect other types of efforts, is based on motivating and then enabling people to do what is right for your organization. What is right will of course vary, but this typically includes motivating teams to take advantage of, and to evolve, existing corporate assets following common guidelines to increase consistency, and working towards a shared vision for your organization. Appropriate governance is based on trust and collaboration. Appropriate governance strategies should enhance the ability of DAD teams to deliver business value to their stakeholders in a cost effective and timely manner. Unfortunately many existing IT governance strategies are based on a command-and-control, bureaucratic approach which often proves ineffective in practice. Chapter 20 of the DAD book provides a comprehensive discussion of agile governance.
8. **Open and honest monitoring.** Although agile approaches are based on trust, smart governance strategies are based on a "trust but verify and then guide" mindset. An important aspect of appropriate governance is the monitoring of project teams through various means. One strategy is for anyone interested in the current status of a DAD project team to attend their daily coordination meeting and listen in, a strategy promoted by the Scrum community. Although it's a great strategy that we highly recommend, it unfortunately doesn't scale very well because the senior managers responsible for governance are often busy people with many efforts to govern, not just your team. Hence the need for more sophisticated strategies such as a "development intelligence" approach supported via automated dashboards.

Ambler has defined two variations on the overall DAD lifecycle model that differ in the development process that is used. Figure 17.1 shows the overall DAD lifecycle model with an agile development process.

Figure 17.2 shows the overall Disciplined Agile Delivery Framework with a lean development process.

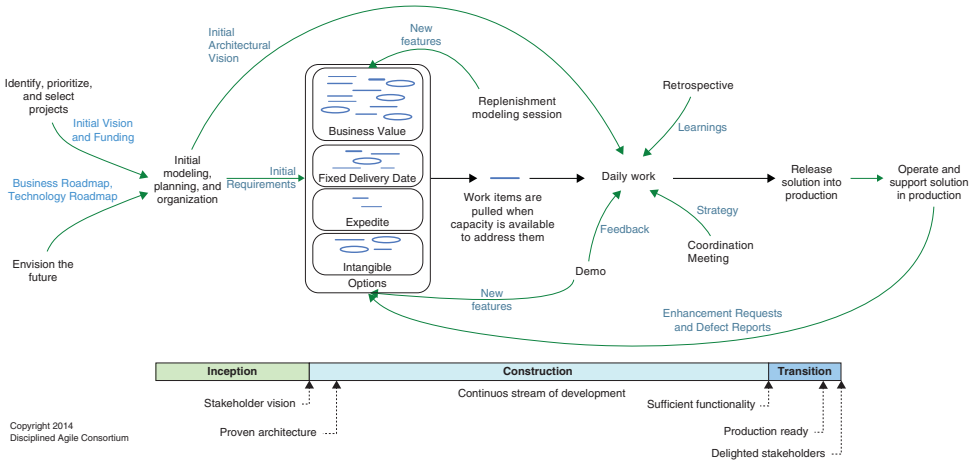


FIGURE 17.2 Disciplined Agile Delivery Framework with lean

SUMMARY OF ENTERPRISE-LEVEL FRAMEWORKS

The three enterprise-level frameworks discussed in Chapters 15, 16, and 17 all provide a way of bridging the gap between an agile development process based on Scrum and the higher-level management practices found in many typical businesses. Table 17.2 shows a brief comparison of the three frameworks.

TABLE 17.2 Comparison of Enterprise-Level Frameworks

Framework	Pros	Cons
Scaled Agile Framework	■ More of a complete top-to-bottom agile approach	■ Requires a more significant transformation to agile ■ May not be appropriate for many companies
Disciplined Agile Delivery Framework	■ Based on a standard agile development process (Scrum) or lean ■ Provides extensions to the agile development process for scaling to an enterprise level	■ Limited to project-level layer only ■ Not a complete enterprise-level framework but is easily adaptable to existing higher-level management levels
Managed Agile Development Framework	■ Hybrid process with a blend of traditional plan-driven and agile approaches ■ Minimizes the level of organizational transformation needed in organizations with a traditional management structure (Could be done as an interim step in an agile transformation)	■ Limited to project-level layer only ■ Not a complete enterprise-level framework but is easily adaptable to existing higher-level management levels

PART 5

Case Studies

IN ANY BOOK OF THIS NATURE, it's always useful to go beyond theory and concepts and show how companies have actually put these ideas into practice in the real world. Of course, there is no canned approach that works for all companies—each of these case studies is different and shows how a different approach may be needed in different situations. It also includes a chapter on "Not-So-Successful" case studies, which shows some of the problems that can develop in an agile implementation.

Chapter 18 – "Not-So-Successful" Case Studies

You can learn just as much or more from situations that don't work very well as you can from more successful implementations. Agile is a very difficult thing to do and is based on the principle of "Fail early, fail often." For that reason, these case studies should be regarded as learning opportunities rather than failures. There's another saying that I really like: "If you have never failed at something, you're not trying hard enough." This chapter contains some case studies of companies that were not so successful in implementing agile.

Chapter 19 – Valpak Case Study

Valpak is an example of a company that has done a major transformation of its whole business around a highly agile approach using the scaled agile architecture. It illustrates how a company has successfully scaled agile principles and practices to an enterprise level and how it has begun to thoroughly integrate agile into many aspects of their business.

Chapter 20 – Harvard Pilgrim Health Care (HPHC) Case Study

Harvard Pilgrim is the only private health plan in the nation to be named #1 for member satisfaction and quality of care for nine consecutive years. HPHC was faced with a major redesign of its architecture to move to a service-oriented architecture, which involved over 200 different projects that had to be completed within a five-year period without disrupting its quality ranking. It was further complicated by the fact that most of HPHC's development resources were outsourced to another company. Because of the scope and complexity of this effort, a hybrid agile approach was needed to provide a blend of agility and control to most effectively manage the overall effort.

Chapter 21 – General Dynamics, UK Case Study

The General Dynamics, UK effort demonstrates how an agile development approach can be applied to a large and complex government program. Because of the need to manage costs and schedules of the overall program to meet government contracting requirements, a hybrid agile approach (DSDM) was used.

18 "Not-So-Successful" Case Studies

YOU CAN LEARN JUST AS much or more from companies that have attempted to implement an agile approach and failed or where it has not been completely successful. *Failure* in agile is seen only as a learning experience and is encouraged—without failure, there probably would not be much learning. Here are some quotes on the subject of failure:

Winston Churchill:

- "Success is the ability to go from one failure to another with no loss of enthusiasm.[1]

Thomas Edison:

- "I have not failed 10,000 times. I have not failed once. I have succeeded in proving that those 10,000 ways will not work. When I have eliminated the ways that will not work, I will find the way that will work."[2]
- "Show me a thoroughly satisfied man, and I will show you a failure."[3]

Joan Collins:

- "Show me a person who has never made a mistake, and I'll show you someone who has never achieved much."[4]

Well-known agile mantra:

- "Fail early, Fail often."

[1] "Miscellaneous Wit & Wisdom," National Churchill Museum, http://www.nationalchurchillmuseum.org/wit-wisdom-quotes.html

[2] Nathan Furr, "How Failure Taught Edison to Repeatedly Innovate," *Forbes* (June 9, 2011), http://www.forbes.com/sites/nathanfurr/2011/06/09/how-failure-taught-edison-to-repeatedly-innovate/.

[3] Laurence J. Peter, "Failure," *Peter's Quotations: Ideas for Our Time* (New York: Bantam Books, 1977), p. 177.

[4] Francesca Rice, "19 Joan Collins Quotes We Wish We'd Said Ourselves," *Marie Claire* (February 4, 2014), http://www.marieclaire.co.uk/blogs/545502/the-joan-collins-quotes-we-wish-we-d-said-ourselves.html. Several unverified sources claim that Albert Einstein said, "Anyone who has never made a mistake has never tried anything new."

The examples given in this chapter are companies that have had problems implementing an agile approach or the implementation has been incomplete. These are real companies; however, naturally, the companies are anonymous.

Agile can be a very difficult thing to do if a significant amount of cultural change is required. It is also a very empirical process, which means sometimes you have to try things to see what works and then make adjustments and corrections (a key idea behind agile is "fail early, fail often"). For that reason, these case studies should be regarded as learning opportunities and not failures. There's another well-known saying that is appropriate here: "If you have never failed at something, you're not trying hard enough." These companies should be applauded for trying to implement agile and, in many cases, they were ultimately successful after an initial false start.

COMPANY A

Background

Company A has a mid-sized IT organization. The company embarked on an agile implementation and trained most of the IT application development staff of about 80 to 100 people in agile practices. An agile coach was brought in for over a year to provide coaching to the teams, and the company made some progress on implementing an agile process at the development team level; however, due to cost-cutting pressure, the agile coach was let go, and there was little or no support at the executive level to take the agile process to the next level.

The approach

About 80 to 100 people in the organization were trained in agile practices, and, at least at a mechanical level, a number of agile practices such as daily standups were being implemented. However, the scope of the effort was very limited to the development organization; the approach was fairly mechanical without an understanding of the principles behind it and without much of an attempt to fit the approach into the company's business environment.

What went wrong

Table 18.1 shows what went wrong.

Overall conclusions

This is a great example of an agile implementation where the company was looking for a "quick hit" and didn't follow through enough to fully develop their agile approach. Agile is a journey and can require some significant organizational change to fully implement it. It is not an all-or-nothing

TABLE 18.1 Company A: What Went Wrong

Problem	Solution
Agile Is Not Just a Development Process	
The company's senior executives saw the agile process as having significant benefits to make IT development go faster; however, they saw it as an IT development process only and didn't see the benefits of investing beyond that level.	Many times companies see agile as a "silver bullet" that is going to make development go faster; they see it as an opportunity for a quick and easy "win" and not as more than a development process. This often results in a partial implementation of agile that doesn't take full advantage of the benefits it can provide.
On the surface, the development effort did *appear* to go faster, but the truth is that people were overworked to make it go faster, and the quality of the products really suffered as a result.	It is often necessary to start out with a limited implementation of agile to get started with, but it's important to set everyone's expectations early on that it is only a start and much more follow-on effort will be required to fully realize the benefits.
Commit Resources to Teams	
The company implemented many of the "mechanical" aspects of an agile development process (e.g., daily stand-ups were held), but there was no real change in the way people were assigned to teams. ■ People were not dedicated to teams and might be assigned to as many as three to four different teams. ■ In some cases, the developers didn't participate directly in the teams and were represented on the team by their managers.	This is, unfortunately, a common practice. Companies implement a few agile practices and call it *agile,* but it really is a very limited implementation of agile. On the surface, it looks "agile" because some of the agile rituals like daily standups are being followed; but it may be only superficial and not really consistent with the real principles behind agile. It is very difficult, if not impossible, to make an agile development process work effectively if the majority of the people on the team are not dedicated to that team and don't even participate directly in it.
This was clearly not consistent with a true agile approach. Without dedicated people on teams, it was almost impossible to stabilize the velocity of the teams and accurately predict performance.	The problem in many cases is that the company sees agile as a development process that only impacts the IT organization; focuses on the mechanical implementation level without understanding the principles behind it; and never follows through with a more complete organizational transformation to make it really work.
Change Is Essential	
There was no fundamental change in the way the company handled requirements: ■ A separate product management group was responsible for producing a business requirements document (BRD) and handing it off to the development team.	The company operated in a highly regulated environment within the financial services industry, and the control being exercised by the product management group had been seen as essential to tightly control and manage the product development effort. However, control is not an "all or nothing" proposition. There are lots

(*continued*)

TABLE 18.1 (*Continued*)

Problem	Solution
■ The process was not very collaborative and it was cumbersome. The product management group insisted on control of the requirements and became a "middleman" between the development team and the business users for clarification of requirement details.	of ways to implement an effective level of control over an agile project without overcontrolling it. The Managed Agile Development framework described in this book is an example.

proposition and sometimes requires a hybrid approach that is designed to fit the business environment. In this case, an agile coach was brought in and was successful in developing a foundation of basic agile practices, but it just didn't go far enough.

COMPANY B

Background

Company B is a mid-sized IT services company. The company has been rapidly developing a software application development business for their clients and the business has experienced significant growth. The company has used a waterfall process to manage its software development projects and bid fixed-price software development projects. The typical implementation of the process consisted of two steps:

1. There is typically a fixed-price effort to do the planning and design phase to define detailed requirements and design specifications for the project.
2. Following the fixed-price planning and design phase, another fixed-price project is proposed for completing the development and testing phase of the software solution.

Recently, Company B experienced significant problems with one of its largest customers. A critical project with this customer was completely stalled. Company B had accepted a fixed-price contract for delivery of the software based on some incomplete requirements and it wasn't making any significant progress, because Company B's development team was largely idle waiting for requirements to be further defined by the customer.

The customer had made a commitment to complete an initial installation of the software in several months and recognized that it needed to take charge of the situation to get the project moving. The customer replaced their project staff that was directing the project and brought in a new project manager and a number of business analysts to accelerate the requirements definition process.

The approach

After the customer replaced their project staff, the customer gave an ultimatum to Company B that the initial installation deadline still had to be met in spite of the earlier delays in completing the requirements. The customer then laid out a plan consisting of a number of sprints that it wanted Company B to meet in order to hit the initial installation date. The following are some of the most significant characteristics of this effort:

- Both the customer and Company B recognized that a more agile approach was needed to make progress; however, neither Company B nor the customer had any significant experience with implementing an agile software development approach.
- The methodology that was ultimately implemented was not really agile or waterfall—it was really just a brute force effort to get the work done in order to hit the deadline. It was similar to breaking up the overall project into a series of *mini-waterfalls* each being about two weeks in length that were called *sprints*.
- The customer created a development schedule for completing the project based on what it thought was needed to meet the installation schedule and broke up the functionality into sprints based on their estimates of the level of effort. Company B's development team was not directly involved in those estimates.
- Company B was pressured into making a fixed-price commitment for completing the project by the scheduled delivery date with performance penalties for missing the deadline based only on a very high-level understanding of the requirements.
- The business analysts that were brought into the project by the customer to accelerate the requirements definition effort worked with the customer's business stakeholders to create use cases and user stories to document the requirements and there was a very limited amount of direct communication with Company B's development team during that process.
- Prior to the beginning of each sprint, the business analysts who represented the customer turned over approved requirements documents in the form of use cases and user stories to Company B to be implemented in the next sprint.
- A limited amount of integration testing and user acceptance testing was included in each sprint due to the time pressures to get the work completed on time, and some time was reserved at the end of the project for doing final integration and testing prior to release.

What went wrong

In situations like this when there is a project failure, there is a tendency to take a brute-force approach to just put pressure on the situation to make the project work, rather than getting down to the root cause of some of the problems and taking a more systemic approach to address the core issues that

tend to make any project successful. In this case, there were four systemic issues that needed to be addressed, which are summarized in Table 18.2 in the following four general areas:

- Project Governance
- Process
- People
- Tools

In the end, the project turned into a "Death March" project to meet a firm delivery date that had been committed to with very incomplete requirements and, that in itself, was very problematic.

Overall conclusions

This is a great example of a project that was failing, and because of the time urgency of meeting a deadline, a brute-force effort was initiated to get the project moving without taking the time to take a more systemic approach to address the core issues that were causing the project to fail.

This is also a perfect illustration of the need for an agile project management approach. In this particular situation, a pure agile approach would not have provided much confidence of meeting the dates the project had to meet and a hybrid approach was needed that provided some level of predictability and control over the costs and schedule of the effort blended with a more agile approach for further elaborating the detailed requirements as the project progressed.

It also indicates the importance of a collaborative spirit of trust and partnership between the customer and the service provider to break down barriers to allow the project to work much more efficiently based on direct communications rather than an arm's-length contractual relationship.

TABLE 18.2 Company B: What Went Wrong

1. Project Governance	
Problem	**Solution**
From a business management perspective, Company B had not adequately recognized the risks and uncertainties associated with software application development projects and had not developed an effective business management approach for managing those risks and uncertainties. The company wanted the business with this customer very badly and very aggressively over committed to meet a fixed date with incomplete requirements.	Company B needed to redefine their management approach for managing software application development projects; however, given the time required by the customer to complete this particular project, it was not possible to do a reset and have a significant impact on the approach for this particular project. As a result, Company B had to take a brute force approach to get it done without a well-defined methodology. A better solution would have been to take an incremental approach to improving the management process as the project proceeded. That approach would have consisted of using an agile approach to identify and prioritize potential areas for improvement and implementing the most critical actions that could be done without significantly disrupting the project as it was in progress.

TABLE 18.2 (*Continued*)

1. Project Governance

Problem	Solution
From a project management perspective, Company B had a PMO that was used for managing other types of projects but the PMO was not heavily involved in software application development projects and had no project managers who were trained and experienced in managing software development projects. A senior-level solution architect and a technical director were used to manage the effort for Company B. A project manager was assigned to play a supporting role to handle some project administration and reporting.	This is a great example of how the need for effective project management in a software development effort is often overlooked. The software development effort was seen primarily as an effort that needed to be led by senior-level developers—project managers were seen primarily as administrators who were more heavily associated with plan-driven, waterfall-style projects. Company B did not fully understand the concept of agile project management and how to better define the project roles to support an agile project management approach. In order to develop an effective approach, the roles in providing overall project management needed to be much more clearly defined. In an agile project some of the project management functions are distributed among the members of the team; however, in this particular situation, the team was not at that level of maturity; and, even if it was, that wouldn't necessarily eliminate the need for a defined project management role in managing projects of this nature.

2. Process

Problem	Solution
Because there was no alternative to Company B's waterfall-style process for software application development, in any situation where that process doesn't work or isn't acceptable to the customer, the fallback was to use no process at all or be at the mercy of a customer-defined process from customers who are not sufficiently experienced to provide that kind of direction.	This is a classic case of where it is perceived that it is an "all or nothing" choice between a totally planned and controlled Waterfall approach and a totally unplanned and uncontrolled approach. A pure agile process would not have worked in Company B's environment as it would not provide a way of setting and managing customer expectations for the cost and schedule for completing projects. A hybrid agile approach was needed that provided a way of managing customer expectations combined with a sufficient level of flexibility and adaptivity to define the details of requirements as the project was in progress.

(*continued*)

TABLE 18.2 (*Continued*)

2. Process	
Problem	**Solution**
Somewhat of an adversarial relationship had developed between Company B and the customer because each side blamed the other for the earlier project failure. That made it difficult for Company B and the customer to develop a joint approach that was optimized to make the project successful.	In order to make this kind of approach work, it is essential to develop a collaborative spirit of trust and partnership between Company B and the customer to jointly manage expectations about the project.
3. People	
Problem	**Solution**
A more agile software development process was needed but that is heavily dependent on having highly skilled and well-trained people—it also can require a considerable shift in thinking and sometimes there is resistance to that kind of change.	In this situation, training of Company B's people was badly needed. However, a standard agile training course would have had limited effectiveness. An ideal solution would be to first better define how a hybrid agile approach would work and then provide training in the context of that approach; however, it wasn't practical in this particular situation to take that approach.
Attempting to do a project like this without people who are well-trained in implementing the project methodology is not a reliable, repeatable, and scalable approach.	A more pragmatic approach in this situation would be to incrementally implement process improvements and training as the project was in progress.
4. Tools	
Problem	**Solution**
In a fast-moving effort like this, tools can be essential for coordinating the efforts of the project team as well as tracking and reporting progress.	The may not be the most critical aspect of a solution for this situation; however, implementation of tools is one thing that can be done fairly easily and phased in without significantly disrupting the progress of the project and it can have a big impact by providing improved communication and visibility into project progress.
Because of the joint nature of the effort between Company B and the customer, the tool needs to be capable of sharing information openly and easily. However, that requires a spirit of transparency and openness for Company B and the customer to freely share information about the project.	Before a tool can be effectively used to share information freely and openly, there needs to be a collaborative spirit of trust and partnership between the customer and the service provider.

COMPANY C

Background

Company C is a company with a small IT organization that has been in the primary mode of supporting existing legacy applications. The company has not had to develop a major new application for a number of years. The existing legacy applications evolved gradually and were developed incrementally over a long period of time. The company initiated an effort to replace and redesign a large, existing legacy application and decided to use an agile approach for the development effort. The company had no previous experience with agile, and agile coaches were not engaged to provide training and mentoring of the project team.

The approach

The company implemented an agile development process from the "bottom up" within the development organization. The process was limited to a development process only, the business participation in the process was limited primarily to JAD sessions to define the requirements, and the role of the product owner was not fully implemented.

What went wrong

Table 18.3 shows what went wrong.

Overall conclusions

It is very difficult to transform an entire company overnight from a traditional waterfall approach to an agile approach.

- Many times, it is appropriate to take a bottom-up approach and start with implementing an agile development process without attempting to transform the higher levels of management in the company; however, when that is done, you shouldn't ignore the higher levels and leave a void in those areas that *isn't filled at all*.
- A hybrid approach such as the Managed Agile Development process is many times a good way to integrate an agile development process with a more traditional higher-level management framework as a first step until those higher levels can be addressed and transformed to a more agile approach. It's like training wheels on a bike—learning to ride a bicycle for the first time as a young child can be a terrifying experience; children tend to fall over quite often and have many accidents. Many times, training wheels are needed until the child gains a sense of balance and the confidence to ride the bike without them.

TABLE 18.3 Company C: What Went Wrong

Problem	Solution
Product Owner Role	
The company didn't understand the role of the product owner in an agile Scrum project and used a traditional model for managing requirements. IT was held responsible for the overall success or failure of the project in meeting business objectives, and the business role was limited to providing input to requirements through JAD sessions. When IT takes primary responsibility for a project of this nature, there is a relatively weak focus on defining the business value that the project should produce without a product owner.	In companies that are in the primary business of developing software products, this role is obvious; however, in a company that uses internal IT applications to manage their business, the strategic importance of those applications may not be appreciated, and the role of the product owner (which is really equivalent to a product manager) might not be understood. A Product Owner needed to be appointed and trained to fill that role; however, given the scope and complexity of this development effort, a better alternative might have been to outsource the whole effort rather than doing it internally at all.
Project Governance	
The company did not have a clearly defined governance model of how the project would be governed. As a result, ■ The right people at the right levels were not engaged in making the right decisions about the project at the right times. ■ Direction from different people was sometimes in conflict. ■ Some people were making decisions that they should not have been making. For example, some of the executives in the company were making detailed decisions about such things as UI screen designs. ■ Some decisions, such as defining the high-level business objectives the project needed to fulfill, were not being clearly defined by anyone at all. ■ The project was way behind schedule with no end in sight and senior executives had lost confidence in the project being successful.	On large enterprise-level projects, there is a need for inputs and decision making from a number of people at different levels, and those inputs need to be organized. A good project governance model should engage the right people at the right levels to make the right decisions at the right times about the project. For example, ■ Senior executives should be defining measurable business objectives that the project should fulfill and delegating more detailed decisions about how the design of the system will fulfill those objectives to others. ■ The managers who are responsible for the business processes should be defining the business rules of how their processes should work. ■ The users and stakeholders who use the system from day to day should have a key role in defining such things as screen designs to ensure that the system is usable. Without clearly defining these roles and responsibilities, there may be conflicts among people attempting to give direction, and some direction might be left out. In this particular case, a clearly-defined project governance model needed to be defined and implemented.

TABLE 18.3 (*Continued*)

Problem	Solution
Development Process	
The company assumed that the development effort could be accelerated by simply breaking up the development process into sprints and using an agile approach for managing the sprints. However, because the agile development process was not fully implemented with the business, the project-level planning and release-level planning that should have taken place was neglected. The result was that the development team was off-and-running developing code, but there wasn't a clear plan for how that code would be released and what the minimum functional requirements for a production release would be.	For large, complex enterprise-level projects, an agile process cannot be implemented only at the development level; there has to be some higher level planning processes associated with it or it is not likely to be successful. ■ Project-level planning is necessary to define a roadmap at a high level for how the product will be rolled out in releases. ■ Release-level planning may also be needed for how the features will be allocated to releases. In many cases, it is also essential to document a high-level plan to define what assumptions have been made about how the project will be rolled out so that there is consensus and buy-in to that plan from all appropriate stakeholders.
Quality Assurance Testing	
There were no formal QA testing resources on the project, and whatever testing was done was done on an ad-hoc basis by developers and business analysts without any formal QA test training. This can be a problem for many agile projects. It was assumed that formal QA was no longer needed, and people on the project such as developers and business analysts who did not necessarily have any formal QA training would perform testing. The result was that testing was very ad hoc without a plan and without well-defined, repeatable test cases to ensure an adequate level of test coverage was provided.	Testing is a science, if it is done properly, and requires people who have some skill in developing well-designed test plans and test cases. An agile development process does not totally eliminate the need for that. A large portion of the testing effort can be done by developers and business analysts, but there is still a role for formal QA testing, especially on large, critical enterprise-level projects. Rather than having a separate QA department perform that function, testers can be integrated into the team, but whoever performs that role on the team should have the appropriate testing skills.
Architectural Planning	
In this particular project, there was a significant risk associated with the cutover of the existing legacy system to the new system that needed to be planned.	One of the agile principles is, "Best architectures and requirements emerge from self-organizing teams." This project illustrates how that principle needs to be reinterpreted at an enterprise level. In this particular case, the risks associated with

(*continued*)

TABLE 18.3 (*Continued*)

Problem	Solution
■ There were some significant architectural decisions associated with how the two systems would coexist with each other for some period of time for the transition to be successful. ■ The architectural planning associated with that transition was not given a sufficient level of focus, and a solution to this architectural problem was deferred until well into the development process.	these architectural decisions were so great that they needed to be addressed and a solution planned early in the project. In this situation, none of the developers on the team had the level of expertise required to do the level of architectural planning necessary. Expecting the development team to perform this function without a sufficient level of focus and expertise just isn't realistic. A separate work stream might need to be created that is staffed by people with the right level of experience and focus to do the architectural planning in parallel with the primary development effort in the team. In many large enterprise projects, particularly ones that require multiple teams, a separate team is responsible for architectural planning and direction.
Project Management	
This is an example of a project where a hybrid approach is needed to blend some amount of traditional project management with an agile approach. In this particular project, a project manager was assigned; however, the company tried to implement a pure agile approach at the development level without a higher level of planning to perform some traditional project management functions such as planning a roll-out strategy, developing milestones, and performing general risk management tasks.	In more mature agile teams, some of these project management functions might be performed within the team, but they become especially critical on large enterprise-level projects. This is a perfect example of the need to fit the methodology to the project rather than force-fitting the project to a pure agile approach. It is important to make an assessment of the scope and complexity of the project and develop an approach that is appropriate to the project. In this particular project, a hybrid approach such as the Managed Agile Development approach is probably needed to provide a blend of traditional project management at the macro-level with a more agile development process at the micro-layer.

TABLE 18.3 (*Continued*)

Problem	Solution
Company Culture	
The culture in this particular company was very sales-oriented and also very heavily focused on operational excellence: ■ The individual users had sales goals that they needed to meet, and there was a significant amount of pressure to meet those goals. ■ The management approach in the company had a strong command-and-control orientation. As a result, there was a lot of top-down direction to the project without a sufficient level of delegation of responsibility and empowerment of the team.	It's very difficult, if not impossible, to change a company culture like that to make it more compatible with an agile development approach, and any change in company culture can take a significant amount of time to implement. The best approach is probably to make the senior managers aware of the impact of these cultural differences and make a conscious decision of how to mitigate their impact. A hybrid approach such as the Managed Agile Development process can be a good way to layer an approach that adapts an agile development process to a culture like this that is not fully compatible with agile.
Tools	
An agile project management tool was used on the project, but no one on the project team was fully trained in its use. As a result, the tool was not well utilized, and it was very difficult to plan and organize the project.	Tools are essential in most cases to manage large enterprise-level projects, and people on the project team need to be trained in their use to know how to use them effectively.

This is also a good illustration of the need for agile coaching and training in implementing an agile transformation. Although the company had some people on the project who were trained in agile and understood the mechanics of how to apply an agile process at a development level, there was no one on the team with a sufficient level of training and expertise to implement an agile process on a large, complex enterprise-level development effort such as this.

19 Case Study—Valpak

BACKGROUND

Established in 1968 and headquartered in Largo, Florida, Valpak is one of the leading direct marketing companies in North America, owned and operated by Cox Target Media (CTM), a subsidiary of Atlanta-based Cox Media Group (CMG). In addition, Valpak has one of the largest collections of digital coupons on the Internet with thousands of local products and services, as well as national brands.

Working in partnership with its network of nearly 170 franchisees in the United States and Canada, Valpak helps more than 54,000 businesses a year to achieve their marketing goals. Valpak's primary competitors are Money Mailer, Valassis (Red Plum), and Super Coups. Valpak's secondary competition includes newspapers, television, Yellow Pages, and any other forms of advertising. All in all, Valpak is trusted by consumers and merchants alike to consistently deliver value. The Blue Envelope® delivers savings and value to nearly 40 million households each month. Annually, Valpak will distribute some 20 billion offers inserted in more than 500 million envelopes. Valpak also offers digital solutions with www.Valpak.com®, an online site for printable coupons and coupon codes, which has nearly 70 million offer views each month, as well as apps for smartphone platforms.

Valpak's IT group builds and supports technology for a wide variety of stakeholders and audiences, including consumers who are focused on saving money with coupons, Valpak franchises that need systems to run their business and sales operations, merchants interested in tracking and maximizing their returns on investment, and traditional internal corporate stakeholders that need to run the core business operations. Efforts for these distinct audiences include:

- To some companies, it may simply mean moving from a waterfall-style development process to adopting a more agile development approach.
- To others it may mean transforming the way the whole company operates. An example is America Online (AOL). Years ago, AOL used to be a dial-up modem company, but as the Internet technology changed, AOL recognized that it needed to transform the very nature of the company to survive.

It needed to shift from being an Internet service provider offering dial-up modem access to a fast-paced provider of media content in order to shift its value proposition to continue to attract subscribers. That required changing the way the whole company operated, in addition to implementing a much more agile process to rapidly develop new media content. They used an overall agile transformation driven top-down by their CEO to transform the whole company to focus on delivering very-high-quality media content to the market quickly.[1]

- Some companies may be far along the agile adoption curve, trying to move on to the next level; and
- Some may be just starting out.
- We were able to successfully build a partnership with the government client in which we did a very professional job of managing overall contractual requirements at the macro-level.
- Within that *macro-level envelope,* we were still able to implement a fairly agile development approach at the micro-level.
- Increasing the level of predictability and control requires beefing up the macro-level, providing more detailed requirements at that level, and implementing at least a limited amount of change control.
- To increase the level of agility, you can simply eliminate the macro-level altogether or limit it to only very high-level requirements.
- Other elements of the framework can be easily customized or eliminated depending on the scope and complexity of the project and other factors.
- The standard implementation uses a model very similar to Scrum for managing the development effort, which uses a prioritized product backlog to drive the development process, and the work to be done is broken up into iterations.

A more advanced implementation uses a lean approach, eliminates the iterations, and replaces them with more of a pull approach that allows work items to be addressed whenever there is capacity to work on them.

- *Valpak franchises*: Order entry, office management, mobile/online sales tools, and CRM applications are developed to support Valpak's 170+ franchise locations located in the United States and Canada. These franchises are independently owned and operated locations that utilize Valpak's franchise system to sell Valpak print and digital products to local merchants.
- *Consumers*: Savings/coupon applications and websites are developed that provide daily value to consumers looking to save money on their purchases. Consumers can interact with Valpak savings *anywhere at any time,* regardless of whether they are using Valpak's traditional "print" mailer or one of Valpak's several digital channels (web, mobile, SMS texting, e-mail). Valpak's savings content is also distributed to over 150 partner websites as well.

[1] Jochen (Joe) Krebs, Presentation to the Agile Boston Group, February 2010.

- *Merchants*: Online websites and mobile applications are developed to allow merchants to manage their advertising campaigns with Valpak.
- *Corporate*: Traditional back-office operations, including manufacturing, marketing, finance/accounting, order processing, and sales. The IT group develops and supports various ERP and custom application solutions to automate these back-office operations.

Valpak's ability to utilize technology to transform their business is a very significant factor in their business success, and Valpak's IT group is an integral part of the business transformation and growth of the company. To compete with the quickly changing digital savings marketplace, Valpak transitioned the entire IT organization to agile Scrum/Kanban processes with two-week sprint delivery cycles. They embraced this change and quickly adapted. This effort was so well done in the IT organization that Valpak is now driving the agile culture throughout the company, heading toward "The Agile Enterprise."

As director of agile leadership at Valpak, Stephanie Stewart[2] has been responsible for leading the agile transformation. In this role, she is responsible for process facilitation, portfolio governance, program management, project management, and of course, oversight of related people, processes and tools. Stephanie leads the team of agile project leaders, who handle everything from project management to Scrum mastering to leading Kanban teams. A self-admitted agile enthusiast, Stephanie has worked passionately to encourage and support the IT organization at Valpak in fully embracing agile software development, to move Valpak toward a greater vision of "The Agile Enterprise."

Chris Cate, CIO, is the agile executive champion working with Stewart and the executive leadership team in transforming the company over the past year. Cate also evangelizes "The Agile Enterprise" vision by encouraging the adoption of agile values and the use of agile methods for non-IT departments.

Bob Damato, director of software engineering, led the adoption of agile technical practices such as test-driven development, continuous integration, and evolutionary architecture. Strong leadership for technical practices across teams has been critical to maintaining and improving quality as part of the agile transformation.

OVERVIEW

Valpak's overall enterprise-level approach is based on the Scaled Agile Framework,[3] which is shown in Figure 15.1 (see Chapter 15). The Scaled Agile Framework (SAFe) consists of three primary layers:

1. *Portfolio layer*: The portfolio layer is the highest and most strategic layer in the Scaled Agile Framework, where programs are aligned to the company's business strategy and investment intent.

[2]http://www.linkedin.com/pub/stephanie-stewart/31/317/a01.
[3]http://scaledagileframework.com/.

2. *Program layer*: The Scaled Agile Framework recognizes the need to align and integrate the efforts of multiple teams that are engaged in large, complex enterprise-level development efforts to create larger value to serve the needs of the enterprise and its stakeholders.
3. *Team layer*: The team layer forms the foundation of the Scaled Agile Framework and is where the fundamental design, build, test activities are performed to fulfill the development requirements for each major area of business. At Valpak, there are actually two different development processes that are used in the team layer, as shown in Figure 19.1. In most cases, Scrum is used for more exploratory development, while a Kanban process is used for *run the business* kinds of development.

Valpak implemented the SAFe from the bottom up:

- In October 2011, Valpak started with six Scrum teams and three Kanban teams at the agile teams (bottom) layer of the SAFe (Valpak currently has 10 Scrum teams and 3 Kanban teams).
- Shortly after the agile teams were established, Valpak implemented road-mapping and release management with the middle layer of the Scaled Agile Framework in mind.
- Most recently, Valpak implemented the portfolio Kanban at the top layer of the Scaled Agile Framework with their leadership team of executive sponsors.
- Last but not least, Valpak added the architectural Kanban.

There are two key things that are most significant about this case study:

1. Valpak recognized the need to adapt the agile development process at the team level into an overall enterprise model that is well integrated with their business. Valpak is one of the initial pioneers in the use of the Scaled Agile Framework to provide that integration.
2. Valpak also recognized the need to use a Kanban process for *run the business* efforts, instead of Scrum, which is used for exploratory development, because the needs are very different.

Both of these efforts show real thought leadership to fit a methodology (or combination of methodologies) to the business and projects instead of force-fitting the business and projects to a predefined, textbook approach.

Architectural Kanban

The *architectural Kanban* is a recommended practice in the Scaled Agile Framework (SAFe) that was implemented by Valpak. It recognizes the need at an enterprise level to plan and implement consistent and well-integrated architectures across all teams and projects.

This approach defined by the SAFe consists of defining architectural epics, which are "large technology initiatives that are necessary to evolve portfolio solutions to support current and future business needs."[4]

[4]Dean Leffingwell, "Architectural Epic Abstract," http://scaledagileframework.com/architectural-epic/.

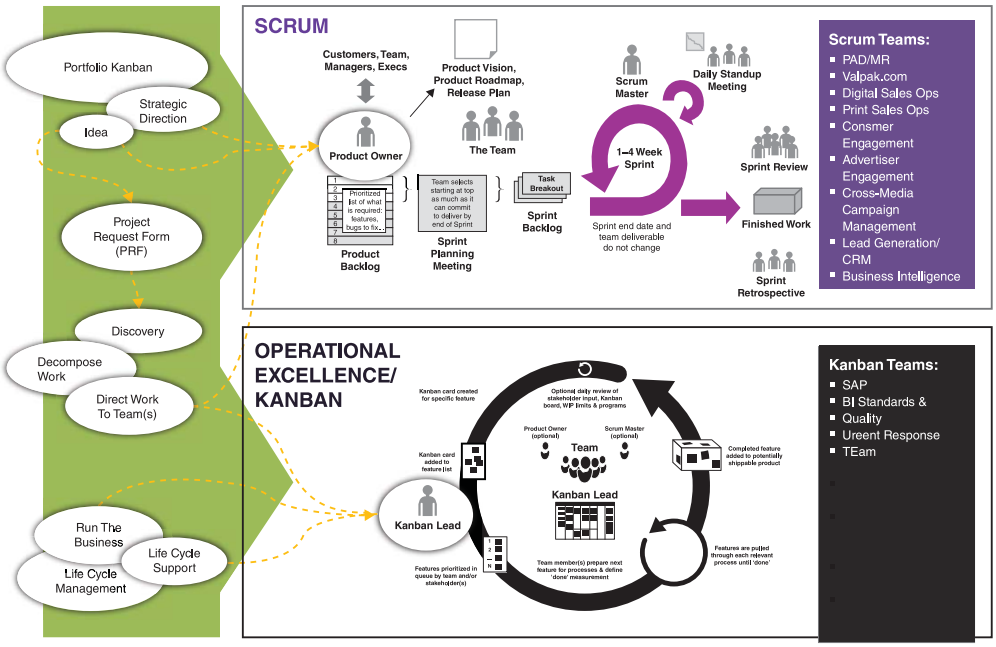


FIGURE 19.1 Valpak agile process framework

According to Leffingwell, sources of architectural epics can include the following:

- Mergers and acquisitions, which require technological integration
- Technological change and infrastructure obsolescence
- Performance and scalability challenges of existing solutions
- Cost and economic drivers, such as avoiding duplication of effort

Leffingwell says these are "initiatives of epic proportions," because they typically cut across three dimensions:

1. Time: requiring multiple PSIs to implement, perhaps taking up to a year or two to complete
2. Scope: affecting multiple products, applications, and solutions
3. Organizations: affecting multiple teams, programs, business units, and even external entities"[5]

At Valpak, architectural epics are captured in the architectural backlog, which is part of the architectural Kanban system. They are processed through various states of maturity until they are moved to implementation. According to Leffingwell, the following are the primary motivators for using this architectural Kanban approach at Valpak:

- Make the Architectural Epic backlog and ongoing analysis visible to all.
- Provide WIP limits to ensure the architects and teams analyze responsibly, and do not create expectations for implementation or timeframes that far exceed capacity and reality.
- Help drive collaboration amongst the key stakeholders in the business and Development Teams.
- Provide a quantitative, transparent basis for economic decision-making for these most important technology decisions.[6]

Table 19.1 shows how the architectural Kanban board is organized at Valpak.

[5]Ibid.
[6]Ibid.

TABLE 19.1 Architectural Kanban Board

Swim Lane	Value Stream Stage					
	Queue	Research	Design	Prototype	Development	Done
Scalability						
Performance						
Reliability						
Technology Upgrades						
Frameworks & Infrastructure						
Innovation						

The "Value Stream Stage" indicates the stage of progress of each architectural Kanban epic ("Development" often involves the Scrum and Kanban teams helping to implement the architectural epic for their area of concern).

The "Swim Lane" rows in the Kanban chart indicate how the architectural epics are grouped by area of focus.

The architectural Kanban board is made highly visible to all teams and provides work-in-progress (WIP) limits on each of the stages in the Kanban process. Weekly standups are held with the architects to review progress and discuss any issues.

Portfolio Kanban

The *portfolio Kanban* is a recommended practice in the Scaled Agile Framework that was implemented by Valpak. The purpose of the portfolio Kanban is to provide a way to plan, prioritize, and manage a portfolio of business epics. "It brings visibility to upcoming work as well as work in process, helps facilitate product development flow and can be a key factor in achieving enterprise—as opposed to team or program—agility, and thereby more fully optimized business outcomes.

According to Leffingwell, the Kanban system is used in this context to accomplish several purposes:

- Make the strategic business initiative backlog (upcoming business epics) fully visible.
- Bring structure to the analysis and decision making that moves these initiatives into implementation, and make that process visible to all.
- Provide WIP limits to ensure the teams responsible for analysis do so responsibly, and do not create expectations for implementation or timeframes that far exceed capacity and reality.

- Help drive collaboration amongst the key stakeholders in the business, Architecture and Development Teams
- Provide a quantitative, transparent basis for economic decision-making for these, the most important business decisions.[7]

The implementation of the portfolio Kanban at Valpak included:

- Highly visible (physical) Kanban board; however, no WIP limits were applied at the portfolio level because it wasn't found to be meaningful.
- Portfolio Kanban standups held with executive sponsors weekly. Each executive sponsor addresses any board movement of epics and any major decisions made.
- Definition of an epic:
  - ≥ 3 sprints of effort
  - ≥ 3 agile teams to coordinate
  - Considerable corporate, franchise, or market value/impact
- The portfolio Kanban is reset at the beginning of each new quarter based on the output from an executive quarterly planning meeting called the Portfolio Review Board (PRB) and based on retrospective improvements identified. PRB focuses on planning only; status and day-to-day collaboration is left to the portfolio Kanban process.
- Value stream (includes entrance/exit criteria) on portfolio Kanban includes *funnel, vet, design, build, deliver,* and *done. Build* is usually when the epic is being developed by the Scrum or Kanban teams.
- The *funnel* is divided into "current quarter," "next quarter," and "unplanned," so that everyone knows what epics have been planned for and what epics have come up that were not in the last quarterly plan. Epics not in the quarterly plan are subject to dot voting.

Challenges with the portfolio Kanban have been:

Standup derailment risk: There is too much talk about strategy or decision making or solutioning during the standup. The director of agile leadership facilitates them; without that facilitation, they tend to fall apart.

Executive schedules: Executive schedules are always busy, so not every executive sponsor shows up every week. Valpak tries not to cancel portfolio Kanbans if two or more executive sponsors are available.

Definition of an epic: There was initially some confusion over what an epic was and if it deserved to be on the board (versus a strategy or a feature or a story). Valpak recently settled on a definition of an epic for this purpose.

[7] Ibid.

Tying strategies to epics: A list of all active business strategies has been established to tie to the epics on the portfolio Kanban. The goal is that strategies spawn one or more epics, and epics are managed via the portfolio Kanban process.

PROJECT MANAGEMENT APPROACH

Valpak has implemented the Scaled Agile Framework (SAFe) very closely. The SAFe does not recognize the concept of a "project"; however, Valpak does have a handful of projects wrapped around large cross-team efforts. For those projects, Valpak assigns an agile project leader to work with the relevant Scrum Masters, product owners, and stakeholders to pull it all together, but there are only about a handful of these projects running at any given time. Most work (even work that crosses multiple teams) can be managed by collaboration among the product owners and Scrum Masters for the agile teams.

At Valpak, projects are represented by epics, and each epic has an executive sponsor. Epics are managed via the portfolio Kanban process (described earlier). The portfolio is planned and prioritized quarterly using these epics. In most cases, epics require the work of three or more Scrum / Kanban teams to accomplish and therefore have more than one product owner involved.

Some epics also have important business tasks to be managed as well (not just a technology effort). For this kind of large epic, Valpak assigns an agile project leader as a project manager to coordinate the work of the business and the agile teams. A waterfall type of schedule may be applied to a project (which is typically the case when you have to fold in the business tasks); however, it is still an agile project management approach overall.

Where project management is needed for large cross-team efforts, the agile project leader will create a project plan (schedule) or a roadmap to represent all the teams, sprints, and business tasks involved with the effort as well as significant milestones. Even though project management may be applied, the agile values are still top of mind for the agile project leaders. This might mean a light roadmap versus a lengthy project plan; in other words, the agile project leaders plan and document to the needs of the project at hand. They apply the degree of project management that produces the most value and expect that the plan is certain to change.

Valpak uses four levels of planning:

1. *Daily*: Scrum team standups in front of task boards
2. *Biweekly*: Scrum sprint planning (all teams run on a common sprint schedule with sprint planning on a Monday and the sprint reviews on a Friday, two weeks later)
3. *Monthly*: Release planning across Scrum teams (product owner-level planning with a six-month look-ahead)
4. *Quarterly*: Quarterly plan of epics (executive sponsor-level planning that fuels the portfolio Kanban for the upcoming quarter)

Valpak has learned to fit the project management methodology to the type of project:

- Scrum is used for most exploratory development and currently consists of 10 teams aligned with each of Valpak's major areas of business focus.
- Kanban is used for run-the-business type of work and/or areas that are not conducive to or don't require the highly prescriptive nature of Scrum.
- Infrastructure projects and large-scale corporate projects are managed as waterfall with some of the Scrums and Kanbans producing work that supports those projects.

According to Stewart:

> There is no one-size-fits-all methodology; at least not that I know of . . . Project Managers need to be more versatile and flexible than ever before. PMs need to be able to effectively pivot between command-and-control and servant leadership focused facilitators. Even though I love working with agile teams right now, I also know your run of the mill Traditional project when I see it. I've learned to pivot between the two by applying different soft skills. Traditional is all about directing and managing whereas agile is all about serving and facilitating.[8]

Tools, communication, and reporting

The following is a summary of Valpak's approach for tools, communication, and reporting:

- All Valpak agile teams maintain physical boards in a common area. Scrum teams use a task board format to display their stories and tasks and show their progress each day. Kanban teams have boards with their custom value stream, and sometimes swim lanes for further classification of work. Additionally, Valpak uses Pivotal Tracker for Scrum teams to manage its backlogs and sprints.
- To assist with managing cross-team dependencies, each product owner tags stories (in Pivotal Tracker) with noted dependencies on other teams. A report is distributed each week that shows these dependencies so that product owners and teams can coordinate and collaborate accordingly.
- On the day after sprint planning Monday, two reports are distributed:
  - Sprint accomplishments (from previous sprint) and
  - Sprint goals (for current sprint)
- On sprint review day, an e-mail of sound bites of accomplishments for all Scrum and Kanban teams is sent to a broad group of stakeholders.
- After the product owners perform their monthly release planning process at the feature level, a one-page view is distributed with six-month look-ahead.

[8]Stephanie Stewart, comments from "Podcast This Week"

- After the Valpak executives meet for their quarterly planning of epics, a one-page view is distributed with a four-quarter look-ahead.
- After the last grooming session of the sprint, each Scrum Master sends out a quick list of proposed stories, anticipated to be planned into the next sprint. These are highly subject to change but help product owners and stakeholders to see what's coming to better coordinate and collaborate across teams.

CHALLENGES

Cultural and organizational challenges

Table 19.2 is a summary of the key cultural and organizational challenges faced by Valpak and how they were handled.

TABLE 19.2 Solutions to Cultural and Organizational Challenges

Challenge	Solution
Managing Cross-Team Dependencies Valpak has very highly integrated systems. In establishing the agile teams, it was almost impossible to create teams that could operate independently of any other teams. In most cases, teams are dependent on one another for a given feature or epic. The most common dependency at Valpak is between Valpak's BI Scrum teams and all other teams, since just about everything involves the ability to track and report.	Besides continuous cross-team collaboration and communication, Valpak has implemented: ■ Cross-team dependencies report distributed each week based on dependencies tagged in Pivotal Tracker ■ Proposed stories distributed week prior to new sprint ■ A meeting referred to as the "Scrum Powwow" held each week with product owners and Scrum Masters to discuss current sprint, next sprint, and roadmap dependencies ■ Visibly flagging dependencies between teams on task boards ■ Shared acceptance criteria prior to or early in sprint ■ Better planning and coordination of handoffs between teams ■ Allocation of time prior to the sprint for proper discovery and architecture across impacted teams for large/complex cross-team epics ■ Cross-team post-planning standup with the impacted teams the day after sprint planning to sync up stories/tasks and collaborate early

(*continued*)

TABLE 19.2 (*Continued*)

Challenge	Solution
Accountability at the Top	
Since SAFe was implemented bottom up, the agile teams were in place well before the portfolio Kanban. Prior to the portfolio Kanban process, there were major disconnects between the executives and the product owners. Mixed directives and conflicting direction was coming from the top, leaving product owners and, therefore, teams with roadmap whiplash.	■ Executive sponsors named for each Scrum team to establish accountability and ownership at the top ■ Product owners meet regularly with executive sponsors on their upcoming sprint plans and roadmaps ■ Portfolio Kanban established ■ Quarterly planning of epics ■ Weekly executive-level standups in front of portfolio Kanban board ■ Executive sponsors frequent the sprint reviews ■ Retrospectives held with executive sponsors to look for improvements
Product Owner Collaboration	
Collaboration can be just as difficult for product owners as it is for the teams. At first, product owners were very comfortable working within their own teams. However, where there were dependencies with other teams, there were often collaboration issues between product owners.	■ At first, a Scrum-of-Scrums meeting was held for 15 minutes each week in the task board common area with product owners and Scrum Masters to review dependencies and their progress. ■ The Scrum-of-Scrums format becomes difficult beyond five teams, so Valpak evolved to the "Scrum Powwow" format. The Scrum Powwow is a one-hour meeting held each week with product owners, Scrum Masters, architects, and other IT leaders to collaborate on dependencies for current sprint, next sprint, and roadmap across teams. ■ Product owners look for proposed stories distributed each week for dependent teams; product owners have access to one another's backlogs in Pivotal Tracker. ■ A cross-team dependencies report is distributed each week that helps product owners to manage dependencies with other teams.

TABLE 19.2 (*Continued*)

Challenge	Solution
Franchise-Based Company	
Valpak is a franchise-based organization. This means that while products and features may be released more quickly under an agile framework, they aren't necessarily adopted, utilized, or sold any quicker by the franchises. Rolling out new products and features to Valpak's franchise organization takes considerable planning and support.	■ Valpak's sales and marketing organizations are becoming more and more agile to quickly enable franchises with new products and features. ■ Training and communication are happening more quickly and frequently than ever before to keep up with the agile releases. There will always be those franchises that are slow to adopt new products, new tools, and new features; however, Valpak is pushing as hard and fast as possible to roll out new products and features. If the product or feature proves to be valuable, most franchises will get on board.
Managing Stakeholders	
At Valpak, stakeholders are anyone who isn't the team, the product owner, the Scrum Master, or the executive sponsor. This means that stakeholders are at all levels of the organization and all departments across the business. Stakeholders also include Valpak's franchises and consumers; however, those particular stakeholders are represented indirectly by Valpak's sales and marketing organizations, respectively. Product owners had difficulty in managing the various, diverse stakeholders. Each product owner had dozens of stakeholders to deal with, each with their own needs.	■ All stakeholders were given one-hour training on agile, Scrum, and Kanban. ■ Over time, product owners refined their approach to stakeholder management. Each product owner has a slightly unique approach based on the needs of their stakeholders. ■ Some product owners began regular stakeholder meetings with core groups to elicit their needs and keep them apprised of progress. ■ Some product owners require that their stakeholders submit a project request form (PRF) to articulate their stories in proper format.

(*continued*)

TABLE 19.2 (*Continued*)

Challenge	Solution
Agile Culture Shift An organization's culture can't be changed overnight with a simple announcement like "going agile!" In fact, it can't be changed with just a kick-off and some training, either. Starting out with agile back in October 2011, Valpak had about 30% naysayers, 40% indifferent, and 30% enthusiasts.	■ The culture that is *agile* must be constantly developed and nurtured over time with every meeting, decision, action, and event that takes place across the organization. ■ To help implement this culture shift at Valpak, an agile coach was involved (three to four days a week on site) for about eight months. ■ To continue the culture shift, Valpak has agile excellence meetings with product owners, Scrum Masters, and IT leadership every other month to perform a retrospective of agile (a retrospective of retrospectives, if you will). ■ In addition, what was the old PMO was restructured to the agile leadership office, led by the agile leadership director supporting a team of agile project leaders. ■ To continue the culture shift momentum in fun and memorable ways, Valpak holds events with the agile teams and stakeholders like Xbox Kinect Fruit Ninja Tournaments. Most recently, Valpak held an Agile Roast for the first anniversary of agile at Valpak.
Developing High-Performance Teams With any agile transformation, teams start off by "doing agile"; following the basic mechanics without necessarily understanding or embracing the values. Such was the case at Valpak. All teams were trained and knew just enough to be dangerous.	Moving the teams from the basic mechanics of agile to truly high-performing teams; no longer "doing agile" but "being agile." Valpak has recognized that developing self-organizing, high-performance teams doesn't happen automatically and requires leadership and continued support. "Agile is not about getting out of the way of your teams but rather staying involved as servant leaders."[9]

Technical challenges

Table 19.3 is a summary of the technical challenges faced by Valpak and how they were handled:

Other challenges

Table 19.4 is a summary of some other challenges faced by Valpak and how they were handled.

[9]Ibid.

TABLE 19.3 Technical Challenges and Solutions at Valpak

Challenge	Solution
Lack of Continuous Integration (CI) No organization or structure around automated testing. No automated execution of tests.	■ Valpak implemented continuous integration where code is committed throughout the day with tests automatically run and errors sent to all developers to resolve.
Build Process Lack of structure and organization around build process. Build process was completely brute force and highly manual.	■ Valpak's build process is now semi-automated and progressing toward fully automated with twice-daily builds to development and test environments.
Legacy Code Base At the beginning of agile, Valpak had a legacy code base consisting of over a half million lines of untested, unclean code.	Valpak implemented Test-Driven Development (TDD) methodology, along with clean code practices. Percentage of test coverage continues to increase with continuous support of TDD and clean code practices.
Architecture Role/Involvement Early on, the architecture role was not well defined. In addition, as a leftover from the waterfall days, teams did not feel empowered to make architecture decisions.	■ Architectural Kanban ■ Architect and lead developer roles better defined ■ Improved collaboration between architecture and teams
Manual Tests / Quality / Risk No real automated tests existed, which increased the risk of quality issues as Valpak moved faster in delivering working software to the business. Releases required what Valpak calls "production checks" to ensure that everything was deployed correctly.	Valpak is currently in the infancy stages of QA automation using Selenium. Teams have begun to automate tests by including technical stories in each sprint.

TABLE 19.4 Additional Challenges and Solutions at Valpak

Challenge	Solution
Planning Sprints for Sustainable Pace	
Out the gate, no team had proper velocity measures. At first, teams planned too much into their sprints and ended up exhausted in trying to accomplish their sprint goals.	■ Over time, the agile leadership office worked with teams to establish some best practices for keeping within their sustainable pace. ■ Most teams have a pencils-down day built into their ground rules in which no new development can be completed toward the current sprint. Stories may be dropped as a result, but sustainable pace is preserved. ■ All teams use a custom capacity calculator (spreadsheet) that was created to take into account non–sprint work like meetings, administrivia, and support in order to better estimate sprint capacity for each team member. ■ So as not to overcommit to a sprint, most teams use the concept of extra-credit stories. Extra-credit stories are established during sprint planning by the product owner as a stretch goal of sorts; if time and capacity allow, the team will complete them.
Decomposing Stories	
Teams are often challenged to decompose stories to fit within a single sprint and without breaking them down by task (develop, test, etc.).	Proper story decomposition can be more art than science. Through training and continuous coaching from Scrum Masters, product owners and teams learned to decompose stories better with each sprint.
Story Points	
Early into Valpak's agile transformation some teams still struggled with the proper use of story points. Many teams were applying story points as a measure of complexity rather than effort. In addition, some teams were applying hours to tasks and skipping the story point step altogether, which basically negates the benefit of this relative estimating technique.	Through training and continuous coaching from Scrum Masters, teams learned to think of story points as effort. Story points are a relative measure of effort to be done, not complexity, not priority, not sequence. Also, the point of improving Valpak's story point estimating is to someday stop having to think about hours.

(*continued*)

TABLE 19.4 (*Continued*)

Challenge	Solution
Team Collaboration Spaces At Valpak, the software development organization has the advantage of being collocated in the same building. However, as a leftover from Valpak's waterfall days, team members were scattered about, which meant there were no team collaboration spaces.	■ Just a few months after Valpak's first sprint, a massive office cube move was organized across 70+ employees to bring together team members with their teams into cube quad areas. ■ Within the cube quads, the inner walls of the cubes were taken down to create a "bullpen" for each team. ■ To top it off, creative signs were created for each team and hung over their cube areas. ■ With open spaces for each team, collaboration and osmotic communication increased dramatically.
Non-agile Tools ■ Prior to agile, Valpak used a traditional project management tool called @task. So, Valpak's agile process inherited a non-agile tool. This meant that sprint planning and backlog management were tedious and time-consuming exercises for each team.	■ After an evaluation of several agile tools, Pivotal Tracker was selected. Pivotal Tracker was considered just enough tool for Valpak's purpose, remembering the Agile Manifesto value of "Individuals and interactions over processes and tools." ■ Pivotal Tracker did not replace the need for physical task boards, which provide high visibility to the process that a tool can't match.
Predicting Release Dates In some segments of the agile community, it is not considered very agile to predict a release date and be held responsible for meeting that schedule. Because of the nature of the Valpak business, it was essential to be able to predict release dates with some level of accuracy.	Valpak adopted a strategy to predict release dates. " . . . by nature of time-boxing, the schedule is a fixed constraint. With the schedule being a fixed constraint (as is cost based on team size), it is scope that remains flexible. So, the scope that is included in any given release is flexible based on what the team was able to accomplish, what stories get dropped, what natural disaster impeded the team, who was out sick, and so on and so forth. Whether that scope accomplished is acceptable for release to production is always a product owner decision."[10]

[10]Blog, "Release Dates in Agile Not Taboo," posted by Stephanie Stewart, June 25, 2012, http://iamagile.com/2012/06/25/release-dates-in-agile-not-taboo-4/.

TABLE 19.4 (*Continued*)

Challenge	Solution
Developing Agile Project Leaders Valpak recognized the need to define a new leadership role and job description for its project leaders so that someone in this position can assume the role of Scrum Master, Kanban leader, and/or project manager, depending on the work at hand.	Valpak created a new job description for an *agile project leader* oriented around leadership of technology-focused projects and teams relying on agile values and principles. "The focus of this position is on delivering value over meeting constraints, leading the team over managing tasks, and adapting to change over conforming to plans."[11]

KEY SUCCESS FACTORS

Four factors helped make agile a success at Valpak.

Top-down support coupled with bottom-up drive

Taking the agile transformation seriously and supporting it at all levels of the business was essential to make the culture shift to the agile mind-set. Stewart explains:

> Rather than continue to dabble with Agile, we went full-fledged, full-bore Agile (all software development teams all at once). Once our CIO declared that we were going Agile there was no looking back. He gained the support of our executive leadership team and they helped us to sell it to our stakeholders, including our franchise network. At first, there were some hold outs (the resistance) but with each and every sprint we have managed to convert a few more into believers. Whenever someone says, "Agile is not working for us," I like to respond with "No, we are not working for Agile."[12]

Hiring an independent coach

Attempting to DIY (do-it-yourself) agile can sometimes backfire. You need to spend the money on bringing on an agile coach for about six months to train the teams and support the process. Companies often tend to appreciate the advice of an independent expert more than a trusted member of the company's own staff.

[11] Jim Highsmith, *Agile Project Management* (Reading, MA: Addison-Wesley, 2010).
[12] Stephanie Stewart, personal e-mail correspondence.

Continued support each and every day

Where agile has failed with other companies, it seems that it was done with a big one-time kick-off without continued support. Stewart continues:

> - I think I read once that a culture change takes 10 years, so . . . We continue to support agile each and every day. That's my primary role as director of agile leadership, but I am well supported by it directors, product owners, agile project leaders, and even team members that have become great evangelists in their own right. The more you talk about it, the more they talk about it. Ways in which we continue to support agile here are things like:
> - Agile Excellence meetings every other month (the retro of all retros, so to speak)
> - Scrum Powwow each week (cross-team current sprint, future sprint, and roadmap discussions)
> - Team Building (with Xbox Kinect Tournaments for agile teams to compete against one another)
> - Celebrations (we did an Agile Roast for our first anniversary this month, and we have cake a lot for important sprint/release milestones)
> - Agile tours given to special groups of stakeholders or VIPs on request
> - Facility sponsor for Tampa Bay agile meet-ups, along with helping to secure speakers and topics
> - Agile Manifesto and other agile-related posters highly visible throughout the IT work space[13]
>
> Stewart said, "It's fun to hear people dropping the *A bomb* (that is, mentioning Agile) at large-scale events like our annual Coupon U with our franchises or corporate all-hands meetings. Agile is really a rock star around here!"[14]

Senior management engagement/business ownership

Getting Valpak's senior management engaged and committed to the effort was a significant factor in the success of the effort. Chris Cate describes the process:

> Valpak has fundamentally changed the way we get work done. This is not just a new process that only involves technology teams but more of a mind-set across all groups that puts an emphasis on driving real business value to our customers in short iterative cycles.

[13]Stewart, personal e-mail.
[14]Ibid.

Business is deeply engaged with their products and services that they are asking IT to build.

- Business (Exec Sponsor and Product Owner) control the backlog and work output—they own it!
- They now have complete ownership and accountability—no place to hide.
- They are actively engaged in managing priority discussions with their customers to identify the highest valuable feature that the team should work on.
- The business is getting more work done across all teams.
- None of our business owners would ever go back to the older Waterfall process.[15]

RESULTS AND CONCLUSIONS

1. *More strategic management focus*. The agile development process, using empowered, self-organizing teams, has enabled a major shift in the Valpak management approach. Cate explains it this way:

Conversations are fundamentally changing at the Sr. executive team level, focusing less on tactical implementation issues and more on strategic growth initiatives:

- Do these things less
- No longer have to justify and/or explain extremely large project plans (the Executive Sponsor and Product Owner decide the sprint schedules and backlog priorities and own it)
- No longer have to talk about why IT missed a date or did not set expectations correctly between project team and Executive Sponsor (ES and PO own the priorities and expectations)
- Rarely have to play the 'peacemaker' role between various groups now that the business firmly owns the work product
- Started doing more of
- Strategic planning and execution to grow the business
- More involved in business development, partnership, and acquisition executions
- Partnering with other Sr. team members to help them think through growing their business lines versus fixing IT[16]

[15]Chris Cate, Cox Target Media CIO, personal e-mail correspondence, November 12, 2012.
[16]Ibid.

2. *Management of IT resources*. The shift to more empowered, self-organizing teams has also made the management of IT resources much easier. Cate says:

> - Less time needed to manage personnel issues since the process helps to correct this naturally
> - Teams are empowered to solve their own problems through collaboration, thus reducing functional manager activities
> - Much more open and transparent environment where the employees, or contractors, not fully contributing are identified and coached up or out quicker[17]

3. *Time-to-market*. Valpak releases software to production at the end of every sprint. Time-to-market was significantly improved as a result of shorter iterations. With two-week sprint cycles, new/enhanced software was available more quickly than ever before to stakeholders, including internal, franchises, merchants, and consumers.
4. *Alignment and collaboration*. Alignment and collaboration between business and IT were increased as a result of a highly visible process. Product owners became truly accountable for their products and teams, and the "us versus them" mentality quickly disappeared.
5. *Employee productivity and morale*. Productivity and morale among the teams improved as a result of the empowered and self-organizing team's principle of agile. In fact, team morale and pride are greater than they've ever been at Valpak.
6. *Delivering more frequent value to customers*. A value-driven process is now in place. Decisions are made on what to work on based on value—that value is derived from stakeholder input. With two-week sprint cycles, teams deliver the highest value stories each sprint. Value is being delivered to stakeholders more frequently than ever before.
7. *Openness and transparency*. Teams, at the portfolio and architecture levels use highly visible task boards to track progress—anyone on the team or in the company can easily see a visual status of what stage of work each story / task is in at any time. In addition, teams know what they're going to be working on for the next two weeks, and all the teams have visibility into what the other teams are doing.
8. *Responsiveness and adaptivity*. Changes can be implemented more quickly than ever before. It is easier to change strategy and pivot. This makes experimentation and risk taking much more feasible. Being able to try out new ideas and turn on a dime is critical, not only for startups but also for larger companies that do not want to become obsolete.
9. Software quality. There is increased emphasis on Test-Driven Development (TDD), unit testing, and clean code. Continuous integration allows recognizing the impact of a change quickly.

[17] Ibid.

LESSONS LEARNED

Forming projects around teams

Prior to Valpak's agile transformation, when a project was initiated, a team would be formed around the project. In most cases, the team members were already assigned to other projects. Stewart says:

> So, with project plans changing as they do, a team member would typically end up being pulled in multiple directions, reporting to multiple masters, across many projects all at the same time. From a people perspective, this is not a fun place to be (remember, "multitasking" is the new four-letter word). The result was low morale from all the project whiplash going on.[18]

With the new agile approach, Valpak shifted to "forming projects around teams." There are 10 Scrums and three Kanbans. Stewart says:

> We have naturally experienced a shift in our project initiation process. Now, when a project is requested, we evaluate the request based on the existing Scrum and Kanban teams, and *we form the project around the teams.* Sometimes this means breaking up the project to fit the vision and/or skill of the teams. So, as opposed to forming a team around a project, we are now forming a project around teams.[19]

Planning team capacity and developing a sustainable pace

Valpak's Scrum Masters needed a better approach to refine team capacity calculations during sprint planning and to allow for non–sprint activities (such as product support) that might require some of the team's time during a sprint. Valpak developed and implemented a team capacity calculator (spreadsheet) for this purpose, which factored in the amount of time each team member was actually available to work on sprint activities. Stewart describes the process:

> During the planning session, we mitigate this risk by having team members predict their personal sprint capacity (50%, 80%, etc.), taking into account non-sprint meetings and a predictive measure of support. Also, we can further manage this by only working

[18]Blog, "An Agile Light Bulb Moment: Forming Projects Around Teams," posted by Stephanie Stewart, April 20, 2012, http://iamagile.com/2012/04/20/an-agile-light-bulb-moment-forming-projects-around-teams/.
[19]Ibid.

> on support that our Product Owner deems immediate work. Any other support can be backlogged and planned into a future sprint. Now, should all hell break loose in the support arena, some stories may be dropped or some stories might not meet the team's definition of done.[20]

Using sprint reviews and "science fairs"

There was a lot of difficulty associated with scaling sprint reviews to an enterprise level with multiple teams needing to hold sprint reviews with some of the same stakeholders concurrently.

> Our first take at Sprint Reviews was pure madness! Each ScrumMaster would schedule a separate 1- to 2-hour meeting on the same Friday for each Scrum team. That's nine 1- to 2-hour meetings within an 8-hour business day [—] and on a Friday no less![21]

Valpak developed a common sprint review process accompanied by a "science fair" approach.

> Reviews [were] shortened to 90 minutes, with each team presenting for just 10 minutes each. Within those 10 minutes, the Scrum Master provides a condensed version of the Sprint accomplishments and metrics, and the team demonstrates only the *sexy* stuff . . . Following that 90-minute period, we hold a one hour Science Fair in the cube areas where the teams sit. Just like parents viewing student experiments at a school Science Fair, this hour is for stakeholders . . . to make the rounds at their own pace to each team they have an interest in for any given sprint.[22]

[20] Blog, "Accountability to Sustainable Pace and Work/Life Balance," posted by Stephanie Stewart, April 10, 2012, http://iamagile.com/2012/04/10/accountability-to-sustainable-pace-and-worklife-balance/.
[21] Blog, "Our Evolution of Sprint Reviews," posted by Stephanie Stewart, June 13, 2012, http://iamagile.com/2012/06/13/our-evolution-of-sprint-reviews/.
[22] Ibid.

20 Case Study—Harvard Pilgrim Health Care

BACKGROUND

HARVARD PILGRIM HEALTH CARE (HPHC) is a full-service health benefits company serving members throughout Massachusetts, New Hampshire, Maine, and beyond. HPHC's mission is "To improve the quality and value of health care for the people and communities we serve."[1] For over 40 years, HPHC has built a reputation for exceptional clinical quality, preventive care, disease management, and member satisfaction and has consistently rated among the top plans in the country.

HPHC has a strong reputation for being customer-focused and having a commitment to excellence. HPHC was ranked the number-one private health plan (HMO/POS) in America according to an annual ranking of the nation's best health plans by the National Committee for Quality Assurance (NCQA) for 2012/2013.[2] HPHC is the only private health plan in the nation to be named number one for member satisfaction and quality of care for nine consecutive years.[3] Harvard Pilgrim Health Care of New England, HPHC's New Hampshire affiliate, is the top-ranked plan in New Hampshire.

Summary of HPHC's business:

- Not-for-profit health plan, based in Wellesley, Massachusetts
- 1,100 employees across eight locations
- Over one million members in Massachusetts, New Hampshire, Maine, and beyond
- Full range of health insurance choices, funding arrangements, and cost-sharing options

[1](2011 HPHC Annual Report).
[2]NCQA's Private Health Insurance Plan Rankings, 2012-13.
[3]NCQA's Private Health Insurance Plan Rankings, 2011-13, HMO/POS. NCQA's Health Insurance Plan Rankings 2010-11—Private. *U.S.News*/NCQA America's Best Health Insurance Plans 2005-2009 (annual). America's Best Health Insurance Plans is a trademark of *U.S.News & World Report*. NCQA The State of Health Care Quality 2004.

Key business challenges faced

- Emerging requirements for complex benefits (flexibility)
- Increasing populations and complex processing push the extremes of capacity (scalability)
- Intensifying demand for information and results of processing (transparency)
- Escalating need for faster deployments of new products (speed to market)
- Aging application suite not up to the challenge

Michael Hurst[4] has been responsible for leading the agile project management adoption effort for HPHC. Hurst is the director of the HPHC PMO, which is in the fifth year of a hardware, software, and "project-ware" migration of its core systems for health care and claims management. He started in IT as a programmer for the aeronautical engineering department at MIT 45 years ago. Subsequently, he became a multistate licensed psychologist and an associate professor at Boston University School of Medicine for 25 years. He founded five health care and health IT companies, with one going public.

Vijay Bhatt,[5] deputy CTO of HPHC, has been the agile executive sponsor working with Hurst and the joint project management office (JPMO) in implementing this overall program over the entire five-year period.

Deborah Norton, CIO, has been the supporting executive sponsor for the entire IT strategy program, including the Scrum program track. She also has encouraged the use of Scrum for business projects and for technical but non-IT strategy projects. A key directive in the adoption process was her declaration that all IT strategy projects were to be agile Scrum by default with exceptions made only as requested and approved by her office.

OVERVIEW

HPHC adopted the following IT strategy principles in 2008 to deal with replacing its entire operational software and hardware systems:

1. Create strong business operations and IT partnerships.
2. Develop flexible business architecture to support change (manage risk with component solutions, implemented during a five-year window).
3. Partner with vendors interested in our outcome.
4. Implement component solutions for health care, such as
    - Oracle Health Insurance (OHI) for claims processing
    - Access and Identity Management (AIM) for security
    - Oracle E-Business Suite (EBS) for finance, sales, billing, service

[4]http://www.linkedin.com/pub/michael-hurst/0/541/909.
[5]http://www.linkedin.com/in/vbhatt.

- Edifecs for EDI platform capabilities
- Other application vendors as appropriate

5. Introduce agile approach and philosophy without disrupting the four years of NCQA #1 standing as the best health care plan in America.[6]

Implementation of this plan required HPHC to totally revamp its internal systems infrastructure to move from a legacy big-box hardware/software environment that was no longer supportable to a totally new service-oriented architecture approach, as shown in Figure 20.1. This effort involved the replacement of the majority of the company's most business-critical software applications. The company set a goal to complete this entire effort within a five-year period by the end of 2012. However, new federal mandates and business priorities resulted in an extension into 2013 for the full scope of capabilities. The flexibility of the agile Scrum process allowed us to do this while ensuring that the highest business priorities always were at the top of the development and funding queue.

It was recognized early on that the traditional waterfall approach couldn't deliver results in that timeframe and would have an unacceptable level of risk. The key drivers for change were:

- Business users didn't want to wait an extensive period of time between requirements and UAT to have something delivered.
- The technical team wanted much greater business input all along the way—they didn't want to discover at the end of the project that they didn't get it right.

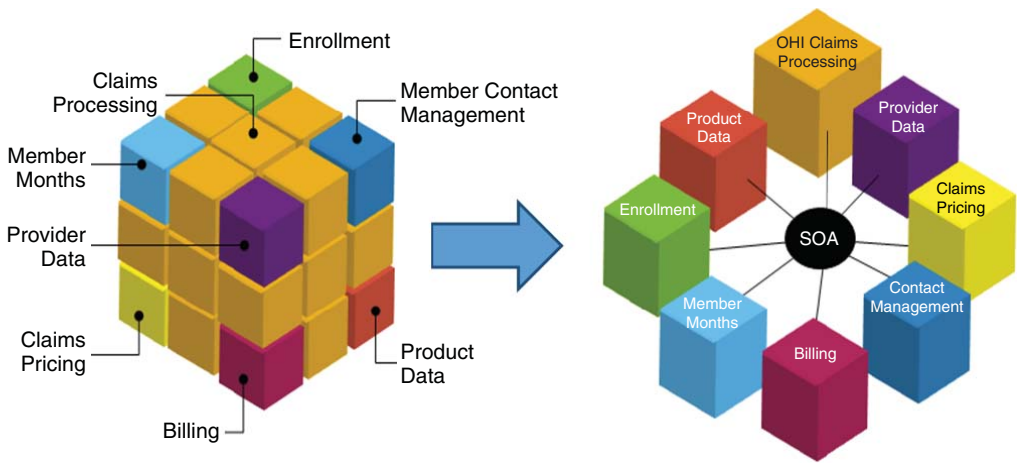


FIGURE 20.1 HPHC architectural transformation

Courtesy of Harvard Pilgrim Health Care

[6]https://www.harvardpilgrim.org/portal/page?_pageid=213,56268&_dad=portal&_schema=PORTAL.

- It was too risky without frequent and continuing business feedback given 10 strategic programs spanning 14 functional areas across five years and a very significant financial investment.
- It was too slow to respond to change in a dynamically changing environment.
- It was too focused on process steps when the need was to ensure delivery of the highest business priority capabilities.

As a result, HPHC developed an incremental and iterative approach—the Scrum framework for an agile transformation across the whole company. All projects associated with this major initiative, called the *IT strategy,* were, by default, to move to an agile development methodology. The new approach that was developed was a hybrid of Scrum at the team level and a more plan-driven approach at the enterprise level to provide program/project management of the overall effort. The total effort involved over 200 different projects that all needed to be well integrated to successfully achieve completing the overall business objectives.

There were a number of cultural obstacles to overcome in implementing the agile transformation. Initially the cultural issues and resistance to an entirely new way of managing projects overwhelmed the intensive training that was received, and a couple of the initial six major projects failed in their first attempts at creating and executing using agile Scrum processes. However, those challenges have been overcome with continuing training and emphasis, resulting in the overall IT strategy being on track for completion with planned business priority-driven extensions. The implementation, which itself has been iterative and incremental, has been very successful in spite of the far-reaching impacts of all the new development and applications.

Impact of outsourcing and vendor partnering

The effort was further complicated by the fact that a large portion of the responsibility for HPHC's IT infrastructure maintenance and application development is outsourced to Dell Services (formerly Perot Systems). New SOA application development, as opposed to developing extensions to the legacy monolithic application or maintenance and updating work, was a new role calling for new skills along with increased resources. Some of these were provided through off-shore resources and some through in-shore resources at Dell. The additional skills, coordination, communication, monitoring, and overall collaboration were a challenge that took considerable time and effort to achieve. The two companies really applied the agile principle of "Customer collaboration over contract negotiation."[7]

In addition, Oracle Healthcare Systems played a significant role in providing a large amount of application software developed in collaboration with HPHC's Scrum teams to fit HPHC's needs. A number of other application vendors also were involved, and a working relationship that meshed with the HPHC environment had to be devised for each one.

[7] www.agilemanifesto.com

As a result, in addition to the normal challenges of an agile transformation of building cross-functional collaboration across different functional departments, HPHC needed to build an equivalent level of cross-functional collaboration among these different companies.

Role of the PMO

Prior to the agile transformation, HPHC had built a traditional, plan-driven PMO organization implementing waterfall software development life cycles (for IT projects). Even before the company moved to a more agile approach, many of the traditional PMO artifacts and processes fell into disuse. It was recognized that a major shift was needed in the role of the PMO. As a result:

- *The PMO in today's organization is relatively small*. It plays more of a consultative role on the methodology and provides training to the rest of the organization on the methodology. It truly has a "servant leader" role as opposed to a directive, command-and-control role.
- *Most of the project managers are distributed into the operational business units with which they are associated*. These project managers are not part of the PMO organization. Only the director, five enterprise-level project/program managers, two senior consultants, a reporting analyst, and a training coordinator are part of the current PMO organization.

Responsibilities of the PMO include:

- Cultural education in the Agile Manifesto principles and philosophy
- Both team and individual training in Scrum practices
- Selecting tools and providing training in how to use the tools for team purposes
- Coaching Scrum behaviors, reinforcing best practices; supporting collaborative learning mechanisms
- Facilitating portfolio, program, and project schedule and resource planning and conflict resolution
- Training and implementation of Kanban release management process
- Training, implementation, and reporting of cross-team dependencies (1,200+)
- Reporting on portfolio programs and projects to all levels of the organization

A joint PMO (JPMO) between HPHC and Dell Services (formerly Perot Systems) was created to deal with the agile rollout, training, coaching, and overall implementation of the IT strategy. The objectives of creating a joint PMO with Dell were to:[8]

- Reduce redundancy, cross-messaging, and project management costs
- Promote by team example, increase communications, share work

[8]Michael Hurst and Hope Krakoff, "Measuring Effectiveness of Project Management: HPHC and Perot Systems," Presentation to the Babson College, Center for Information Management Studies, Waltham, MA October 21, 2009.

- Ensure project success of five-year IT strategic portfolio
- Support new software development process

Project governance

In addition to the JPMO, several other project governance mechanisms were implemented to provide for project coordination and consistent implementation of the methodology, and to provide a framework for ongoing, continuous improvement. The project governance framework included:

1. Product Owners' Forum (the POF), where all business and technical product owners meet with the CIO to coordinate their individual efforts. The POF was chartered with:
   - Improving product owner communications about team progress, hindrances, and cross-team dependencies
   - Identifying opportunities for inter-team collaboration, sharing of resources, development needs and efficiencies
   - Identifying and resolving potential mismatches or conflicts in resources and needs
   - Escalating to IT Strategy Committee for review or ratification if needed
2. Scrum Masters' Forum, where all Scrum Masters meet to coordinate their individual efforts:
   - Issues and answers about the Scrum Master role
   - Tools for meetings and progress tracking
   - Resources and training
   - Shared best practices
3. Business Analysts' Forum, where business analysts meet to discuss:
   - Issues and answers about the business analyst role on agile/Scrum teams
   - Shared experiences and problem solving
   - Resources and training
   - Shared best practices

Figure 20.2 shows the structure of the overall Project Governance Model that was implemented by HPHC. The three listed business unit columns are only examples of the 14 IT strategy initiative business units that were actually in play. There were 10 IT strategy initiatives, 14 business units, and 40+ Scrum and hybrid agile teams; in addition, there were usually 40 to 50 other corporate-level projects being conducted at the same time as the IT strategy ones.

At the end of the second and third years of the agile implementation, a survey was implemented to assess the effectiveness of the methodology and how completely and consistently it was being implemented. The survey was administered to all members of 46 teams in flight that were using agile/Scrum. The survey content was created by Vijay Bhatt and the joint PMO, modeling some of the items on the Nokia survey but modifying as appropriate for our focus on compliance with the

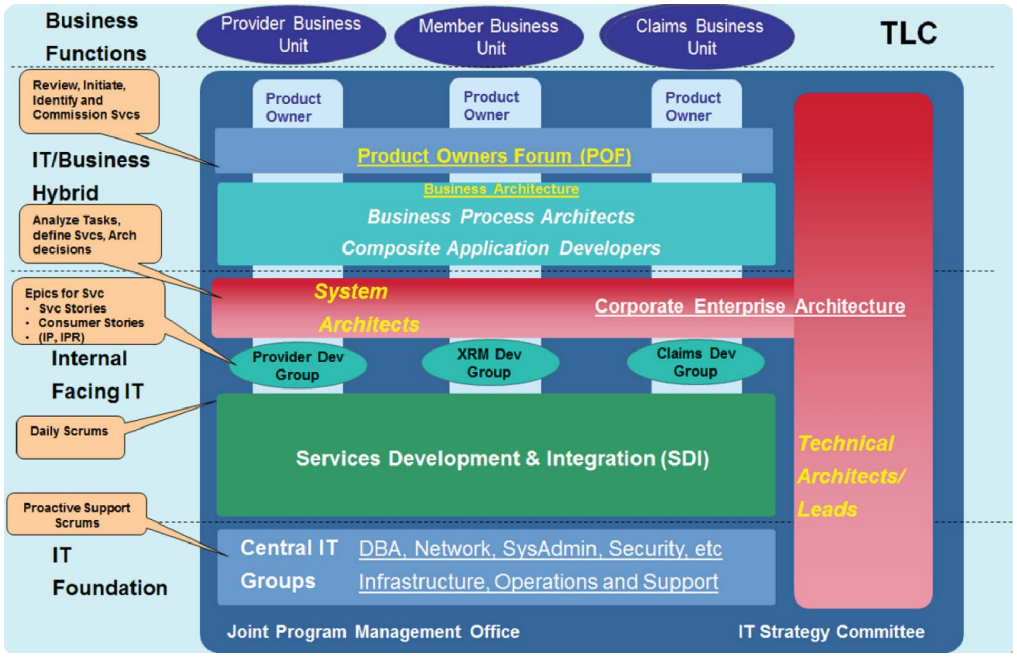


FIGURE 20.2 HPHC IT strategy governance structure

Courtesy of Harvard Pilgrim Health Care

Scrum methodology.[9] Bob Sullivan, senior PMO project manager consultant, conducted the survey and analyzed the results:

- The 2009 results across the many different facets of a Scrum framework showed that these teams achieved 80 to 90 percent compliance in 75 percent of the teams and to a much lesser degree in the other 25 percent (ones that were much more hybrid or even traditional).
- A repeat of the survey in 2010 (obtained in 2011) showed essentially the same results, with some improving and others dropping elements of the Scrum framework.
- Other, less-structured surveys of the product owners and the business units showed a high degree of satisfaction with the new applications and functionality meeting their needs. The IT department also received far fewer requests from the business units for modifications of the functionality they received. A defect tracking program with a daily Scrum was put in place and continually reviewed with the business and development teams to improve the process even further.

In 2012, HPHC had some perceived loss of Agility and Scrum compliance at the team level. Although it hasn't been deemed necessary to do a formal survey again, the HPMO estimates that one-third of the 100+ total project teams are agile Scrum; one-third are hybrid; and one-third are traditional. This kind of variation in project methodologies stems from the widespread types of projects ranging from IS upgrades and maintenance through application configurations on up to full new development projects.

Role of tools

HPHC recognized early on that the scope and complexity of this effort could not be successfully accomplished without the appropriate tools to manage the effort. Initially, as suggested by the Scrum trainers and agile coaches, teams used spreadsheets and story wallboards. However, as the number of teams grew, the complexity of the work and coordination grew exponentially. The simple approach did not work from the portfolio management and reporting perspective, though it was fine for the individual teams until they had to deal with the tremendous number of inter-team dependencies. Nor did the team-to-team Scrum-of-Scrums work. There were over 1,200 inter-team dependencies identified. After a selection process, HPHC selected Rally software and now relies heavily on it to provide overall management and coordination of dependencies, reporting, and integration of the entire effort. The Rally software is used to do the following:

- Provide teams with a common tool for tracking their stories and tasks in a Scrum framework.
- Track and manage interdependencies among teams.
- Provide overall status reporting at both the enterprise level and the project/team level. Teams can use paper-based story cards within their teams if they want to, but all teams are required to use the

[9]Maarit Laanti, Outi Salo, Pekka Abrahamsson, "Agile Methods Rapidly Replacing Traditional Methods at Nokia: A Survey of Opinions on Agile Transformation" *Information and Software Technology* 53 (3) (March 2011), PP. 276–290, http://dl.acm.org/citation.cfm?id=1937326.

Rally software for managing user stories and milestones that have inter-team dependencies either as predecessors or successors. Some larger programs have created federated projects in Rally that contain all the stories and milestones from multiple teams that have external dependencies but have the individual teams managing everything else via wallboards, spreadsheets, Microsoft Project or other tools.

- Provide a Kanban board management approach for monthly promotion to production from all the teams.
- Provide a Kanban team management approach for environment development and test data management.

Project methodology mix

The company has a range of different kinds of projects—most are agile, either Scrum and/or Kanban, but there are also some plan-driven projects.

- Projects in the Corporate Information Management (CIM) group (ETL projects, report development projects, and the like) are managed with their own highly structured and predictable process and tools. They maintain a single Rally *project,* which is the repository of all CIM stories with milestones, deliverables, and dependencies with other team projects.
- The product group (insurance products, not software products) also has a highly structured and predictable path to creating their product deliverables. It has largely traditionally structured projects, though it, too, is now finding that agile might be the better process. As a result, these projects have become hybrid projects.
- At the other end of the spectrum are projects doing new code development for new or existing but incomplete applications. These are all agile Scrum projects.

Also agile (but using Kanban rather than Scrum as the primary implementation framework) are those building environments for testing groups of applications (called *development integrated test* environments) and those creating sets of test data for the different applications, projects, and environments. Both of these teams are highly limited by capacity in staff, hardware, and software that have to be brought together in usable stacks and then rearranged as needed. The Kanban approach fits them very well.

Project portfolio management

Every year, HPHC's senior executives review a list of proposed project initiatives and prioritize and rank those initiatives to maximize the overall ROI of the portfolio. Each project is required to have a one-and-a-half-page Project Opportunity Statement (POS) that summarizes the project objectives, costs, schedule, and expected ROI. Behind this minimum requirement, each project may have a variable level of backup supporting details, depending on the nature of the project. For some projects, a two- or three-page POS may be sufficient; for other projects, much more detailed information is needed to support the request.

The PMO senior consultants provide assistance to the executive sponsors, project leads, and product owners in completing the POS. These are published for community consumption so that teams and the general business community can understand the goals, rationale, and expected outcomes and return to be expected.

The following is a summary of the management approach used to monitor progress against these plans:

- Project managers begin weekly reporting once projects have been approved with a designated budget and staff (HPHC, Dell Services, and other third parties when it is a technical project). Progress against the projected cost and schedule is tracked in Dell's Project Reporter (PR) database reporting tool. It is used for all projects and provides the source of truth for all budgets, resource plans, and progress reports against the approved project milestones and resources. Project and program managers update PR every Friday with basic summaries and detail updates as needed. The PMO then extracts the information and creates an executive dashboard that is published every Wednesday for the leadership team.
- For teams with both a Scrum Master (SM) and a project manager (PM), the PM normally is responsible for the external team reporting. There are some teams where the SM also has the PM responsibilities, normally assisted by a project coordinator or assistant for the administrative duties. There are relatively few PMs who can adapt to both roles at the same time simply due to the external demands, but there are some who have no problem.
- The PMO creates all needed reports from the team entries in Rally and PR.
- Once a month the CIO presents a comprehensive report on the entire portfolio to the executive leadership committee (ELC) and to the board of directors. This comprehensive report is provided as a dashboard along with detailed backup.
- Once a quarter, corporate leadership reviews the status of all the corporate initiative programs and projects in the portfolio. Then the reports are assembled and communicated to the corporate community. Any adjustments in priority or revisions in timelines and milestones are also set at this time as part of the process.

PROJECT MANAGEMENT APPROACH

Project Methodology

The HPHC project methodology for the IT strategy was designed with the following principles. It was to be a(n)

- *Business-based approach*: The projects were not to be just IT projects; they were to be operational improvement projects with business value that met a defined market needs
- *Development approach*: The agile methodology meant business collaboration, increased flexibility, responsiveness, speed to market, and productivity

- *Component (SOA) approach* focused on standards and reuse of services
- *Approach with regular reviews* of capability requirements for course corrections and realignment to the Corporate and IT Strategies over the five-year plan

Figure 20.3 shows a high-level overview of the HPHC lifecycle model. The following is a description of the phases in the lifecycle model.

Implementation package development

Each project goes through a project initiation phase prior to startup to plan the project, which consists of completing an *implementation package* containing the following items:

- Identification of owner and all stakeholders and roles
- Vision, scope, goal
- High-level driving requirements
- Epics (very large stories, also called scenarios)
- Assumptions
- Technology assignments and/or technology guardrails for all services and components

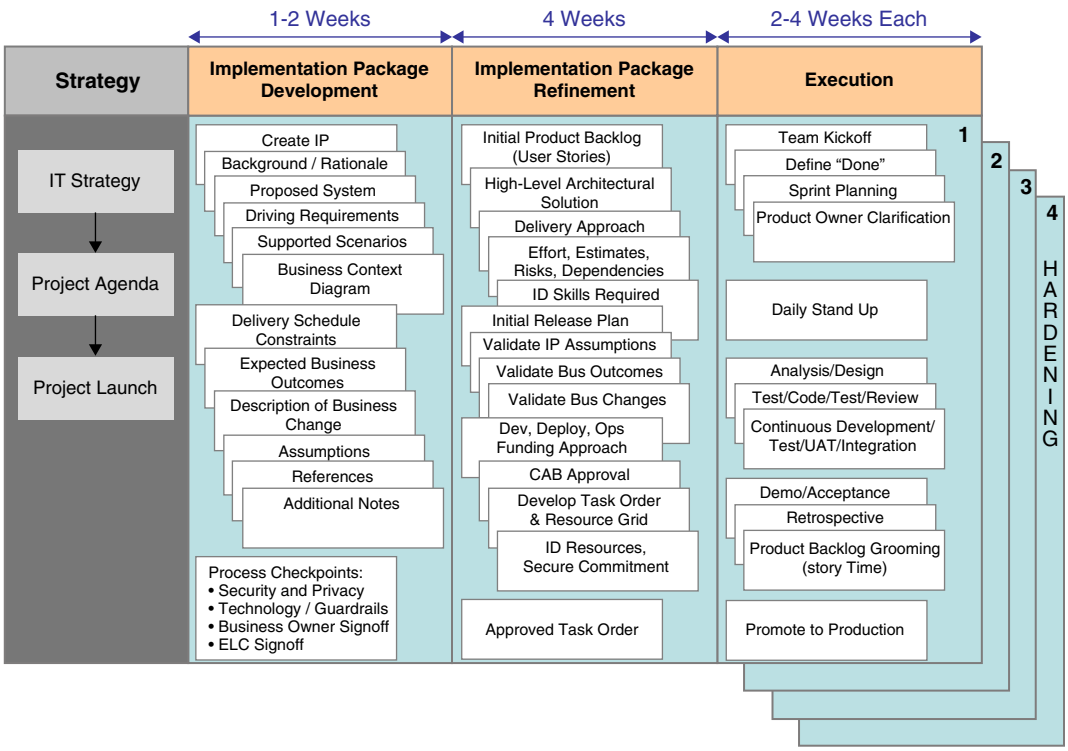


FIGURE 20.3 HPHC project lifecycle model

Courtesy of Harvard Pilgrim Health Care

- Business context diagram (BCD)
- Delivery schedule constraints-flexibility matrix
- Description of the project deliverables
- Recommendation of the implementation methodology
- Expected business outcomes
- Description of the business change
- Privacy and Security review and check-off
- Executive sponsor owner sign-off

Implementation package refinement

After the implementation package (IP) is created in the project initiation phase, the next step is to refine the implementation package:

- The input to this process is the implementation package.
- The output of that refinement is a series of components and, for technology projects engaging Dell Services or other resources, a task order.

Table 20.1 shows a summary of the inputs and outputs that take place in the implementation package refinement (IPR) process:

- The IP is developed by the business, normally a HPHC functional business area, with the assistance of the IT department.
- The IPR process is conducted by Dell Services with assistance from the business unit in developing the product backlog, along with prioritization.
- The entire process takes four to six weeks, except for projects that are a component or continuation of a very large product backlog.

The most successful process HPHC had for agile project initiation was the IP, then the IPR; however, for new teams that did not have an established velocity, both the IP and IPR were limited to only proposing three initial sprints. The team would establish its velocity and could then forecast its release plan (at a high level) for its product backlog with a much better degree of accuracy. After that point, an experienced agile team would use its past velocity in creating subsequent and follow-on projects.

Project reporting

The following is a summary of the reporting tools used by HPHC to manage the project:

- The Rally software was used heavily for project reporting at a detailed ("micro") level, primarily for team reporting but also for dependency coordination and management.

TABLE 20.1 Implementation Package Inputs and Outputs

Inputs	Implementation Package Outputs
Content	■ Product backlog (PBL) ■ PBL Prioritization ■ Approximate PBL sizing ■ High-level architectural solution ■ High-level release plan timeline ■ Expected sprint dates, cumulating in interim releases ("Schedule," "Milestones") ■ Interim releases≫Final release ■ Multi-team coordination within program and PBL ■ Resource grid ■ Release plan
Participants	■ Product owner ■ Technical leads; more of team as needed ■ Scrum Master and/or project manager
Outputs	Task order

- Weekly project status reports were used for executive high-level reporting.

Project Name	Executive Sponsor/ Project Lead	Project Manager	Baseline Start	Baseline End Date	Revised End Date	Product Effective Date	Overall Status
							Green
							Yellow
							Red

- A risk, assumptions, issues, and decisions (RAID) report was also used by almost all teams to track these items and manage their resolution.
- Dependencies between teams' user stories were tracked in Rally, and a dependency report was used to track and report the status of all dependencies.
- Project *dashboard* reports were also updated weekly to provide a high-level status of all projects as shown below:
- A special "Yellow-Red Status Report" was used to report on any project that was in red or yellow status during the daily stand-ups for all 100+ projects in flight at any given time. There simply

were too many participants to allow oral reporting on the standard "What I've done, What I'm doing, and What is standing in my way" Scrum process. Those details were made available through Project Reporter for everyone's review. The Yellow-Red report was used to focus on what was standing in the way for teams. This report did provide the details for just those teams so that the Scrum could focus on resolving blocking issues. For any given Scrum, 1 to 15 teams might be providing their status and their blocking issues. For a team needing assistance or wanting to escalate to everyone's attention, these reports were one of the best ways to do that.

Contractual relationship with Dell Services

One of the most significant complicating factors that HPHC had to deal with was the contractual relationship with Dell Services (formerly Perot Systems). HPHC outsourced a substantial amount of responsibility for maintenance of its IT infrastructure and application development to Dell. In fact, the number of Dell people involved in the effort was significantly larger than the number of HPHC IT people involved. HPHC had about 80 IT people engaged in the effort, along with several hundred from the business units (as SMEs and stakeholders), and another several hundred Dell technical staff, SMEs, and stakeholders from the operating units it maintains for HPHC. In total, about 800 people were participating between Dell and HPHC.

It was essential to develop a close collaborative relationship with Dell to make this work, and both companies decided to form a joint PMO to facilitate sharing of responsibility for the effort. The formation of a joint PMO was initiated to enhance communications and collaboration between the two companies while reducing duplication of effort in a number of areas. Significant results are summarized in Table 20.2.[10]

CHALLENGES

Cultural and organizational challenges

Table 20.3 is a summary of the key cultural and organizational challenges faced by HPHC, and how they were handled.

Contractual challenges

Table 20.4 is a summary of the key contractual challenges faced by HPHC and how they were handled.

[10]Maarit Laanti, Outi Salo, Pekka Abrahamsson, "Agile Methods Rapidly Replacing Traditional Methods at Nokia: A Survey of Opinions on Agile Transformation," *Information and Software Technology* 53 (3) (March 2011), pp.

TABLE 20.2 Results of Joint PMO

Before Joint PMO	After Joint PMO
Dual project reporting systems and data entry	A single reporting system
Two project administration systems (project creation, approval, administration, and change process)	A single project administration, process tracking, and compliance tool
Dual training programs	HPHC management of both internally and externally sourced training for new programs with joint PMO approval
Dual project repositories	Single project repository linking to each PMO's website and the project reporting and project administration systems
Infrequent and informal coordination meetings between PMOs	■ Joint PMO email distribution group for communications ■ Initial biweekly meetings are now weekly to address JPMO backlog ■ Thrice-weekly joint pre-review meetings for new projects and change requests ■ Weekly project reviews ■ HPHC CIO and Dell Client Executive issue escalation meeting

Technical challenges

Table 20.5 is a summary of the technical challenges faced by HPHC and how they were handled.

Other challenges

Table 20.6 is a summary of some other challenges faced by HPHC and how they were handled.

KEY SUCCESS FACTORS

1. *Immediacy and persistency*
   - Addressed issues as they arose (very frequent interactions/follow through to resolution)
   - Individual commitment to success and trust in the other's commitment to success
   - No fear of losing one's job
   - Persisting through trial balloons to useful practices and tools

TABLE 20.3 Solutions to Cultural and Organizational Challenges

Challenge	Solution
Cultural Change	
Many people felt comfortable with a plan-driven approach and had difficulty adopting the shift in thinking that was required to move to a more agile approach. Other cultural issues included: ■ Resistance to change and opposition from employees on both the business and technical side to the change ■ Cultural differences between HPHC and Dell in terms of time and activity tracking needs and requirements ■ Focusing on completion of deliverables rather than process and task steps ■ The "just enough" approach of agile completely butted against the HPHC culture for 120% planning, effort, and thoroughness ■ Territorial (stovepipe) attitudes toward responsibility and collaboration and differences of opinion on resolutions Instead of an HPHC business unit dictating requirements, they had to assign SMEs to work on teams with Dell associates to define and refine stories, acceptance criteria, test cases, and test data. The collaboration was daily rather than quarterly.	■ It was recognized that some people were not going to be able to make the transition to an agile approach, and some of those people moved on to other assignments that were more plan-driven. ■ About 800 total business, IT, and Dell folks were put through training in the first two years; and subsequently all IT and Dell staff routinely take the agile training courses. ■ Special training and coaching are made available to HPHC business units to assist their transition (where desired and possible).
Level of Business Involvement	
Although business leaders wanted much closer involvement in the development of their needed applications, the need for continual work with the team, the reviews, the demonstrations, and the re-prioritizations every two to four weeks with agile teams had a far greater impact on them than expected. ■ The business leaders were very pleased with the results and the delivery time, but in the beginning it was a very foreign process that did not fit in with their traditional day-to-day operations.	■ There is no question in anyone's mind at this point that the more continual involvement of the business with the team (even if only through product backlog prioritization meetings, sprint reviews, and demonstrations) is a far more productive and useful process for both the development team and the business. The business really gets what it wants most in a timely fashion, and that reality has led to acceptance of the additional demands on its time.

TABLE 20.3 (*Continued*)

Challenge	Solution
■ After all, they were not product managers or developers. They were business people running operations with their staff who had to be assigned to these teams as subject matter experts to assist in story writing, requirements, test scenarios, test data creation, and demonstrations.	■ The business has had to really change its allocation of staff time and duties from just their functional orientation to the collaborative team. As many of our coaches and trainers predicted, this transition has taken years, but it is clearly becoming the norm instead of the exception.
Project Portfolio Management	
Agile does not provide any guidance on how to do project portfolio management of a large number of projects like this at an enterprise level.	■ HPHC had to develop an overall approach to determine how projects were prioritized and selected for implementation to maximize the overall ROI of those projects and provide a way of measuring progress once those projects were selected. ■ HPHC also had to devise different levels of reporting that minimally affected teams yet provided the kind of information required at higher levels in the organization.
Scrum Master and Project Manager Roles	
HPHC had to reconcile the Scrum Master role with the project manager role: ■ Those who had training either with HPHC or outside of the company realized that the Scrum Master was not the director of everyone's time and activities but was much more facilitative. ■ Scrum Masters have some activities similar to a project manager role, such as removing obstacles to progress, setting up meetings outside of the team workspace and time, recording progress, and reporting to outside parties. ■ The traditional project managers were more comfortable with their responsibilities for schedule and budget but much less so with the increased facilitative role and decreased command and control.	■ Some Scrum Masters did take on more of the project manager tasks without becoming a "dictator" to the team. ■ Similarly, those who were more project manager-oriented had to be more facilitative, let team members choose and prioritize tasks rather than be assigned them, and empower team members to ask lots of questions for clarifying requirements and tests. ■ The recruiting and selection process of both HPHC and of Dell Services had to change to emphasize individuals with experience and comfort in both the Scrum Master and project manager roles. If individuals only had experience in one role or the other and not both, they needed to provide evidence that they could work with a cohort (either Scrum Master or project manager) in managing a project team.

(*continued*)

TABLE 20.3 (*Continued*)

Challenge	Solution
	■ Some agile projects would have a Scrum Master and a 10% project manager for Dell timekeeping and reporting responsibilities. ■ Eventually, it became more common to have a project coordinator or administrator assigned to handle administrative duties, relieving a Scrum Master to focus on team facilitation.

TABLE 20.4 Solutions to Contractual Challenges

Challenge	Solution
Cross-functional Collaboration Across Companies for Individual Performance Reviews	
Creating cross-functional collaboration across multiple companies (HPHC and Dell and various vendors) was essential because most teams were composed of representatives of both companies.	■ The companies had to agree to change some of their normal performance measurement systems to allow feedback and inputs from outside the company to support a collaborative cross-functional team approach. ■ For both HPHC and Dell employees assigned to teams, a large part of their performance review had to come from the teams, but neither company's HR systems originally allowed such input. Over time, both companies made changes to allow this cross company input and feedback.
Team Assignments and Resource Sharing	
Initially, there were too many shared resources, and with that level of resource sharing, it becomes very inefficient for the people to do "context switching" from one team to another that they participate in.	■ Over time, both Dell and HPHC were able to concentrate team assignments, but it is still a struggle today since all of the Dell staff time is either billed to projects or core maintenance work, while HPHC staff is not accounted for in the same way. ■ HPHC's technology partners, especially Dell Systems, had to change from assigning staff to five or six teams in slices of 16% to 20% time to just one or two teams. However, this is one area that has significantly gone backward, as the pressure on budgets and time accountability has risen.

TABLE 20.4 (*Continued*)

Challenge	Solution
Contracting Approach	
Because the agile development approach relied heavily on outside contractors (Dell and Oracle), a contracting approach needed to be developed that was consistent with the agile development approach. It wasn't possible to force another company involved in a contract to completely change its own project methodology. ■ Standard IT contracting normally specifies a very detailed list of requirements, if not a full work breakdown structure, along with detailed release planning, often months or even years in advance of the work. ■ In this five-year program, that simply was not possible since nearly all the applications were brand-new and were being delivered into a brand-new architecture (SOA). ■ In addition, there was urgency to begin, since both the hardware and software platforms were no longer supported; many new types of features and functions were needed that could not be developed in the old environment, and change was the name of the game in health plan capabilities. Nearly everything was changing with urgency, so the long lead time for detailed prospective planning simply did not exist.	■ The contracts with Dell and Oracle were treated as high-level agreements. It was explicitly agreed and understood that the detailed requirements would be further elaborated as the projects progressed. Broad schedule milestones were set along with very high-level budgets that would be refined as the details became known in the iterative agile process. ■ Some compromises needed to be made to come up with a consistent way of managing the effort at a project level while meeting the overall corporate goals and contract obligations. ■ Both Dell and Oracle, and other vendors, supplied staff to participate locally in teams in addition to their offsite resources. They also supplied coordinators to provide the overall schedule and resource coordination. ■ With Dell, HPHC created the joint PMO. ■ With Oracle Health Insurance, the effort was large enough (10+ projects at any given time) that the HPHC OHI Product Owners created a program office. Then the HPHC joint PMO provided the same kind of coordination and collaboration with that program office.
Compensation, Billing, and Multidisciplinary Roles	
The HPHC reimbursement system paid for job activities according to designated position codes, and Dell staff had to record the activity code along with the project and other information in their timekeeping system. These were contractual terms that were embedded in the financial systems of the two companies. It was a real cultural change for folks to take on multidisciplinary roles and activities.	Since agile Scrum philosophy includes the concept of the team being responsible for completing work and team members assisting others who need additional help, HPHC and Dell Services both had to change their respective time accounting, billing, and payment systems to reflect the fact that developers could write requirements and stories (previously that activity had been restricted to business analyst and subject matter expert job codes). Similarly, business analysts might write test cases, which would have been previously restricted to test architects and QA specialists.

TABLE 20.5 Solutions to Technical Challenges

Challenge	Solution
Service-Oriented Architecture Interdependencies	
The implementation of a service-oriented architecture introduced some significant dependencies between the teams designing the services and the teams developing the applications that consumed the services. The service-oriented architecture required the creation of 14 major services with over 100 interfaces.	The Rally software was used to track and manage interdependencies among teams. In addition there were several forums created for product owners and Scrum Masters to coordinate the efforts of their teams.
Automated Regression Testing	
Because of the fast pace of the effort and the complexity and interdependencies in the overall solution, it was essential to implement some form of automated regression testing as the five-year program progressed.	HPHC used an offshore team as a centralized resource to design and implement an extensive automated regression test capability. Teams now have to provide their regression test scenarios and test data before putting their work into the promotion to production process.
Enterprise-level Architecture Planning and Design	
HPHC has a very complex environment, and it is essential to create an integrated approach for all enterprise systems to work together in a well-designed service-oriented architecture. To achieve that goal, it is essential to have an architecture that is consistently implemented across all project teams. ■ The risk associated with a team developing its portion of the solution in a way that is not consistent with the rest of the architecture and then having to refactor the design and the affected applications to become more consistent is unacceptable. ■ No single project team has sufficient visibility into the overall architecture at an enterprise level to make decisions unilaterally regarding designing its particular portion of the application to be consistent with that overall architecture.	■ All projects go through an architectural review early in the project. ■ An architect is assigned to each team to provide additional guidance and coordination. A SOA Center of Excellence (SOA COE) and Corporate Architecture Board review all projects technical designs and are responsible for approving them and any changes made over time. ■ A formal change management process is conducted through the SOA COE for changes involving dependencies and Services. The process involves all affected teams and in the last year has provided turnaround answers or solutions in less than a week for all cases.

TABLE 20.5 (*Continued*)

Challenge	Solution
■ Some level of upfront architectural guidance has to be provided to the teams, and some level of coordination and review is needed to ensure that the architecture is consistently implemented.	
Software Release Process	
The process for releasing software had to be carefully managed to provide some level of configuration management and release control to ensure that any applications being released were consistent with other applications that were already in production or being released at the same time. Initially, the process for releasing applications was too agile (basically "when ready" according to team standards) and needed to be better controlled.	■ A staged release process was developed to test compatibility of a given application in combination with other applications in different environments prior to release. For example, groups of related applications (ones that "touched" one another) would have an environment created for that level of testing. Over time, a larger systems integration environment was created to allow testing of end-to-end scenarios. Work continues on this very complicated issue. ■ A Kanban process was devised to control the flow of applications through this staging process prior to release. A monthly promotion to production schedule was devised and then used across all teams. In the future, all iterations and releases will be coordinated across all of the teams. ■ As the applications and SOA capabilities became more mature, an incremental process of selectively releasing a limited number of components at a time was adopted that allowed migrating business lines dependent on those capabilities. This approach was adopted, rather than a "big bang" approach of releasing a large number of components simultaneously and trying to migrate the data and records of millions of customers simultaneously. This more iterative approach to migrating to the new environment has proven highly successful.

TABLE 20.6 Solutions to Other Challenges

Challenge	Solution
Office Space	
To provide co-location for the teams, a number of conference rooms were converted into "bullpens" to provide a working area for the teams. This presented a challenge because in most cases the team members had their own individual cubicle plus the shared bullpen. This temporary doubling of space needs presented a challenge because it consumed a lot of floor space unnecessarily until adaptation could be made.	To more efficiently utilize the office space, many of the people who work on agile teams had to give up their individual cubicles and use the agile workrooms as their primary work area. For most on these teams, this approach worked well since HPHC provided "drop-in" cubes for those needing temporary individual space.
Government Regulatory Requirements	
HPHC is required to comply with a number of government regulatory requirements for the healthcare industry. One of those requirements is to use industry-standard procedure codes for all medical procedures. The government has recently required converting from the ICD-9 system of about 1,400 procedure codes to the ICD-10 system of over 140,000 procedure codes. The Department of Health and Human Services (HHS) published a regulation requiring the replacement of the ICD-9 code set with ICD-10 as of October 1, 2014. This is a very ambitious goal with an end-date that cannot slip, and it has a significant impact on most of HPHC's applications.	■ In order to meet this very difficult requirement in the time required, over 40 different agile teams were created, and the effort required to meet this requirement was partitioned among those teams so that they could work in parallel to get this done within the mandated timeframe. This very large mandated program also created its own program PMO to work with the larger JPMO. ■ Many government mandates have to be given priority over other business needs, and the corporate initiative team helps set the priorities and resolve the disputes.

2. *Leadership support*
   - Positive feedback on successes
   - Joint communications announcing implementation of new methods
3. *Well-trained people*.

   Having the right people on the teams who are properly trained in the agile methodology is essential. It is important to recognize that some people will not be able to make the transition, and having alternative teams and work is important for morale in the workforce.
4. *Cultural change*.

   Changing the culture of the organization to enable a more collaborative, cross-functional approach was essential. Some projects will fail if cultural change is not addressed early on and continually reinforced thereafter.

5. *Integrated hybrid methodology*.

   Successfully accomplishing this effort required a combination of an agile approach at the team level and more of a plan-driven approach at the enterprise level.

6. *Tools*.

   The use of tools such as Rally has been essential for managing an effort of this scope and complexity.

7. *Collaborative approach with other companies*.

   The overall project could not have been successfully accomplished without developing a collaborative approach with the other companies involved (Dell, Oracle. and other vendors).

8. *Early planning*

   - Get executive buy-in up front
   - Five-year high-level roadmap (plan for delivery of capability along the way)
   - Fit within an achievable and supportable funding profile
   - Ensure project planning happens to at least some degree in terms of architecture and data management

9. *Execution*

   - Accountability (clear lines of responsibility)—Senior business and IT operations staff as Product Owners
   - Flexibility (don't be afraid to change what's not working)
   - Dependency management (SOA begets intricate dependencies that need significant oversight)
   - Daily but short meetings to monitor and manage progress, and find/resolve impediments

10. *Bringing it home*

    - It took real courage of conviction to begin the migration.
    - The IT strategy is still a work-in-progress, but well over 80 percent of the way there, even when another major corporate initiative had to take priority, extending the five-year program by six months.

CONCLUSIONS

1. Hybrid Project and Portfolio Management Methodology[11]

   - Found a hybrid portfolio approach that was conducive to finding and fixing issues early
   - Delivered value early and all along the way
   - Delivered usable capabilities in year 1 that gave the strategy momentum and credibility

[11]276–290, http://dl.acm.org/citation.cfm?id=1937326 Able Workshop, Bentley University Conference Center, Waltham, Massachusetts, October 12, 2011.

- Enabled a strong business and IT partnership
- Adapted more readily to changing priorities (inevitable over five years)

2. Risk Management
   - Started to cross the bridge before it's finished—very challenging step that pushes everyone in a very interdependent environment
   - Began migration with the simplest accounts to support; then added capabilities just ahead of the migrating accounts (online billing, EDI enrollment, administrative services only billing)
   - Segmented and managed risk along the way
3. Architectural Approach
   - Provided reusable enterprise-wide integration components (services)
   - Extended the capabilities of key applications (e.g., eBusiness suite)
   - Had a corporate architecture board and center of excellence to provide overall perspective and planning for the highly integrated, service-oriented environment
4. Agile Contracting Approach
   - The agile contracting approach was very successful and created strong collaborative relationships with key partners such as Oracle and Dell.

LESSONS LEARNED

Enormous culture shift

There was an enormous cultural shift required for this effort to be successful. Help is needed to make this transition; don't transition to agile without help.

Adapting the methodology to fit the business

- Don't be afraid to adapt the methodology, but use caution with eyes wide open. Agile principles can be applied in many ways.
- Consider the scope and complexity of the effort. As the scope and complexity of an enterprise-level project increases, more planning and structure are needed, and a purely team-based, rather than program, approach may break down.

Release management

Allowing teams the freedom to release to production without synchronizing across dozens of projects with interdependencies among them proved to be very challenging. It eventually required moving to a common release schedule to synchronize capability delivery across many projects.

Assigning projects to teams

The pre-existing traditional process created new teams for every project. As agile became more embedded, it was clear that projects should be assigned to teams, especially teams that had been working any element of the product backlog. This revised process presented more challenges for the outsourced IT vendor, Dell Services, since there was and is less flexibility in assigning resources according to their needs, but it led to much more effective teams.

Preserving the learning associated with teams is important and, for that reason, it is best to keep existing teams intact as much as possible. This may mean fitting the project (e.g., partitioning the product backlog) to existing teams as much as possible rather than reorganizing and reforming teams to fit a project.

Architectural design planning

Agile does not typically involve a significant amount of planning and design planning of the architecture of the system up front. The architecture is expected to evolve and code be refactored as needed as the project progresses.

HPHC found that the costs and complexity of this method were untenable. It was more efficient and gave teams better coordinated direction to have more upfront planning and sign-off of the architectural design approach, with reviews by an architectural board for changes. The level of effort required for reworking and refactoring the design as the dozens of projects progressed for a program of this cost and complexity is both financially prohibitive and disruptive to development.

Estimating project schedules

HPHC developed a lifecycle model that included an *implementation package* (IP) stage followed by an *implementation package refinement* (IPR) stage. In both of these stages of the process, an attempt is made to estimate the schedule for completing the effort. HPHC learned that new teams that did not have an established velocity had a very difficult time estimating a schedule.

For that reason, the IP and IPR stages of the process were limited to proposing only the first three sprints of a project for new teams. After the team gained proficiency and had an established velocity, the team would use its past velocity in creating subsequent and follow-on projects for the same product backlog.

QA testing

The complexity of the QA testing effort at HPHC, with as many as 30 to 40 applications or functional components going into the production environment at the same time, was significant and required a great deal of thought and planning. HPHC needed a new way of thinking about QA and the path to production (e.g., synchronizing promotions of dependent parts) of many different but

interdependent applications (quite different than the software companies who develop a single product or application).

Creating and maintaining the environments for the extensive integrated testing in a SOA environment required a new approach. The costs to create a complete replica of the entire production environment for testing purposes were prohibitive. As a result, HPHC and Dell created the following:

- A team responsible for managing test environments.
- Development integrated test environments that consist of the application to be tested and its immediate neighbors—this is a continually ongoing process.
- A system integrated test environment, which was a subset of the full production environment that would allow end-to-end testing of a variety of scenarios. This environment included all of the applications upstream and downstream (not just the immediate neighbor applications) that were part of the end-to-end scenarios being tested.
- A team responsible for creating test data needed in the environments. It was simply too much to expect project teams to be responsible for the collection of all the needed data, assembling it in a way consumable by the applications in a test environment, and then providing a "reset" capability for continued use. Teams provide the scenarios and descriptions of the data needed to this team, who then get it, assemble it, and proof it with the teams prior to the actual testing work.

Building automated regression testing scenarios and continuing functional capability (updated datasets) to teams is critical to having teams stay productive both in their own environment and in the integrated environments.

CIO retrospective

Cultural change is difficult—far more difficult than we imagined. It is extremely challenging to create an effective case for change when what people have been doing historically has been working and working well. They simply do not see anything it in for them. Identifying the believers at all levels is essential, as is continuous communication and reinforcement of the imperative for change.

The new world order takes a lot of getting used to. Reordering of priorities, reshuffling of resources, and refactoring were not viewed as essential and positive elements historically; rather they were viewed as near failures. Changing the mind-set of "That's not a bug, that's a feature" required senior-level involvement and, on occasion, still does.

HPHC's IT strategy would not have succeeded without the transition to agile. That said, it was important to make the change not with the mind-set of a religious zealot but, rather, allowing

folks to adopt and adapt. That fosters a sense of ownership among the business participants and technologists alike.

A program as large as HPHC's most assuredly requires infrastructure, support, and coordination (and at times direction) from the project management office (PMO). While the product owners and Scrum Masters have significant autonomy, orchestration of this massive effort is critical. The PMO plays a central role in this orchestration. An agile transformation brings tremendous value; portfolio management of an effort this large provides the complementary skills and tools to deliver capabilities effectively.

21 Case Study—General Dynamics UK[1]

BACKGROUND

GENERAL DYNAMICS, HEADQUARTERED IN FALLS Church, Virginia, employs approximately 95,100 people worldwide. The company is a market leader in business aviation; land and expeditionary combat systems; armaments and munitions; shipbuilding and marine systems; and information systems and technologies.

General Dynamics, UK Limited, is a leading prime contractor and complex systems integrator working in partnership with government, military and civil forces, and private companies around the world. General Dynamics, UK has 50 years' experience in technical leadership, manufacturing expertise, and prime contract management skills to deliver C4I communications solutions, armored fighting vehicle (AFV) technology, and security systems for critical national infrastructure and deployable infrastructure. General Dynamics, UK has 11 world-class facilities across the United Kingdom and internationally, with over 1,650 highly skilled members of the team.

Due to the nature of the government contracting environment that General Dynamics, UK has to operate in, projects have to work within two key constraints, time and cost, where other agile methods don't always have this. This case study illustrates how a company can successfully develop a hybrid agile approach that blends together elements of a plan-driven approach for managing time and cost constraints and still provide flexibility to their customers. Nigel Edwards at General Dynamics, explains it this way:

> From a customer perspective, the approach provided flexibility since it allowed the detail of the technical requirement to evolve without the risk of cost and schedule overruns. The

[1] Information provided courtesy of General Dynamics, UK Limited.

approach did, however, require a greater investment on the part of the customer as they had to be an embedded part of the day-to-day working of the delivery team to help ensure the requirements evolved in a way that ensured the end product met the end user's needs. Traditional project delivery approaches allow a more arm's-length relationship.[2]

The DSDM (Dynamic Systems Development Method) framework is a hybrid, project-level agile framework that was successfully used by General Dynamics, UK Limited to deliver the Combat Identification Server (CIdS) Technology Demonstrator Project (TDP) for the Ministry of Defense (MOD) in the United Kingdom. The objective of the project was to help clear "the fog of war" by providing a picture of the position of nearby friendly forces on the ground in the cockpit of an aircraft.[3] The following is a brief summary of some key aspects of the project and is reproduced with the permission of the DSDM Consortium and General Dynamics UK.

Nigel Edwards of General Dynamics, UK was responsible for management of this program. Nigel has 25 years' experience in engineering development projects and 10 years' experience as a program/project manager. Core to this experience has been leading and managing multidiscipline engineering teams and partner company relationships.

OVERVIEW

The following is an overview of some of the key attributes of the approach used by General Dynamics, UK on this project.

Requirements Prioritization and Management Approach

Traditionally, project management in the defense sector has focused on meeting technical requirements, sometimes at the expense of project duration and cost. In a DSDM project, the performance requirements are prioritized and expressed as *must, should, could,* and *won't* (this time) or *MoSCoW*, which is defined as follows:[4]

- **Must have**: These provide the Minimum Usable SubseT (MUST) of requirements that the project guarantees to deliver. This may be defined using some of the following:
- Cannot deliver on target date without this
- No point in delivering on target date without this; if it were not delivered, there would be no point deploying the solution on the intended date

[2]Nigel Edwards, e-mail comments September 11, 2012, and November 2, 2012.
[3]"DSDM Case Study—Improving Outcomes Through Agile Project Management," http://www.dsdm.org/case-studies.
[4]MoSCoW Method, http://www.dsdm.org/dsdm-atern/atern.

- Not legal without it
- Unsafe without it
- Cannot deliver the business case without it

  Ask the question, "What happens if this requirement is not met?" If the answer is, "Cancel the project—there is no point in implementing a solution that does not meet this requirement," then it is a *must have* requirement. If there is some way around it, even if it is a manual workaround, then it will be a *should have* or a *could have* requirement. Downgrading a requirement to a *should have* or *could have* does not mean it won't be delivered, simply that delivery is not guaranteed.
- Should have
- Important but not vital
- May be painful to leave out, but the solution is still viable
- May need some kind of workaround (e.g., management of expectations, some inefficiency, an existing solution, paperwork, etc.)

  A *should have* may be differentiated from a *could have* by reviewing the degree of pain caused by it not being met, in terms of business value or numbers of people affected.
- Could have
- Wanted or desirable but less important
- Less impact if left out (compared with a *should have*)
- Won't have

  These are requirements the project team has agreed it will not deliver. They are recorded in the Prioritized Requirements List *(Scrum's Product Backlog),* where they help clarify the scope of the project and to avoid being reintroduced "via the back door" at a later date. This helps to manage expectations that some requirements will simply not make it into the delivered solution, at least not this time around.

In the DSDM model, trading out requirements provides the flexibility to ensure on-time and on-cost delivery of an acceptable and fit-for-purpose solution rather than a perfect one. The CIdS project requirements were categorized using the MoSCoW approach. Through a trading process during project execution, this technique enabled a few *should* and *could* requirements to be removed from the project solution to prevent potential cost and schedule overruns, without any customer penalties being incurred.

In practice, the development of the CIdS project capability was divided into increments, which provided checkpoints where capability could be demonstrated. Each of these increments was divided into fixed-length time boxes. If, in the delivery of any time box, it was apparent that there was risk to the achievement of the *must* requirements, trade-offs were made to defer requirements from the *should and could* requirements to assure delivery of *must* requirements. If it became apparent that the

must requirements were in jeopardy, then the time box and potentially the whole project could have been stopped or replanned; extension of a time box is not permissible. This discipline is important in the "fail early" principle of DSDM.

Contract Negotiation and Payment Terms

Implementation of this type of approach requires a certain amount of trust between the government contracting agency (or any other customer) and the contractor. The government contracting agency had an initial concern that the contractor would abandon *should and could* requirements at the first opportunity, and an incentivization mechanism was sought. However, this approach contradicted DSDM principles where cost—and hence, price—is fixed, with requirement trading offering the project contingency. During contract negotiation, this incentivization position was relaxed as the government-contracting agency recognized that the contractor would aim to maximize the delivered capability.

One aspect that required ongoing negotiation through the project was the payment plan. Once the project started, it became apparent that the detailed project schedule that was developed during the initial stages of the project no longer fit the payment plan that had been agreed on prior to contract award. By taking a pragmatic and flexible approach and recognizing that this is the nature of DSDM, the government contracting agency and the contractor were able to align the program plan and the payment schedule.

A major challenge was that the initial stages of the project required a fair degree of tolerance and patience with lack of clarity and uncertainty.

Planning Approach

Instead of including a detailed schedule of the project delivery with the bid submission, expert advice was that this process should be performed jointly with the government contracting agency after contract award as part of the project-level planning phase (DSDM Foundation phase) of the project. Before the team launched into detail, it needed to put the project into context by focusing on the fundamental project objectives. This was achieved through the collaborative development of a *Single Statement of User Need*, which stated:

> "Report blue force track information to authorised requesting entities on demand in the Close Air Support (CAS) mission."

As the planning progressed, so did the collective understanding of the technique. A key objective of the DSDM approach is to progressively reduce risk through incrementally delivering a solution. As

such, establishing how the technical solution would be elaborated within the DSDM framework was also a key part of planning the overall strategy for delivery of the CIdS project. Using DSDM, the CIdS project comprised three distinct phases:

1. The *foundation phase* developed the requirements, technical design, and also planned in detail the subsequent exploration/engineering phases.
2. The *exploration/engineering phase* was where the CIdS solution was incrementally developed through a series of time boxes where MoSCoW'd requirements were logically grouped. These time boxes constrained the implementation to time, cost, and quality.
3. The final *deployment phase* demonstrated the CIdS solution to end users.

The high-level plan for the project, laid out during the foundation phase, became the outline that defined the rest of the project. Throughout the CIdS project, focus was maintained on control and metrics regarding achievements within each time box. While all project time boxes were planned ahead at an outline level, as individual time boxes ended, the performance within that time box enabled the detailed planning of the next time box. This incremental and iterative development approach, where achievement informs ongoing plans, was at the heart of the project.

Personnel Management

Selection of the right personnel to lead and work on the project was critical. DSDM requires people with the ability to work effectively within collaborative teams and a tolerance of ambiguity. This was not a project for task-focused, left-brained project managers! Equally the assignment of the technical coordinator role was pivotal; this role requires a combination of technical domain knowledge, foresight, and leadership to ensure the technical solution achieved its requirements.

Communication

Traditionally there is reticence to share "warts and all" information between partners and/or customer. As the team relationship developed, aided by significant periods of co-located working, an open and no-surprises culture of communication developed. A clear no-blame mandate was established to promote open and honest communication. Where frustrations between groups did develop, they were effectively addressed to prevent any damage to the day-to-day working relationships. Other communication aids were also developed by the team.

Through time box management worksheets and clear communication of requirements to be implemented, associated estimates and the allocation of resource were provided. Daily progress of time boxes was reviewed by team leads such that exceptions were managed as they arose instead of later. A key facilitator of effective communication was an electronic shared working environment, supported by e-mail and telephone/video conferencing.

Management and Leadership Approach

CIdS project success was underpinned by the foundation phase outputs and the associated facilitated workshops. Through the foundation phase, the project developed the plans collaboratively, ensuring consensus. Strong leadership, however, ensured that there was no management by committee that could have resulted in indecision and deviation from the project objectives. Establishing the detailed technical requirements and partitioning these into a logical development strategy was challenging, but the detailing of the time boxes was eventually achieved.

Emphasis through the exploration and engineering phase was to ensure that the project objectives were delivered. Managing schedule adherence was relatively straightforward since it was bound by the time boxing. However, metrics were devised to monitor performance against technical outcomes to help inform the planning of subsequent time boxes. These also addressed concerns that incomplete time-boxed requirements could be deferred into later time boxes, potentially resulting in a bow wave—a technical solution only meeting the *must* requirements.

PROJECT MANAGEMENT APPROACH

The overall project management approach was based on the DSDM (Dynamic Systems Development Method) framework. The following is a high-level overview of the DSDM framework. Please refer to the DSDM website for more details.[5]

Project startup: At the start of the project, it was important that everybody involved understood the fundamental principles of DSDM and the mechanics of how to use it as a project management technique. This was achieved through facilitated workshops using a DSDM subject matter expert. Apart from having the DSDM expertise, the added benefit was that the facilitator was neutral, which was helpful when setting up management team structures and the project delivery approach.

Project metrics: Ensuring the project had a measure of delivery performance was essential to ensure the project outcomes could be achieved. Metrics were put in place to measure time-box achievement in terms tasks/requirements achieved, schedule, and cost. While time boxes could not be allowed to overrun, they could finish early. Initially it was important to gauge how well time boxes were estimated to ensure they weren't being overloaded, and to inform subsequent time-box planning. The metrics also provided a mechanism to demonstrate to the customer the project was achieving its objectives.

Team structure: The fact that the industry team had contractual responsibilities could not be ignored and informed the allocation of key roles and responsibilities in the DSDM team structure. The DSDM independent facilitator provided a crucial role in the exercise of allocating individuals to roles during one of the team workshops. The fact that the team was involved in the process helped reinforce understanding of role and responsibilities as well as the team relationships. Developing the team structure

[5]http://www.dsdm.org.

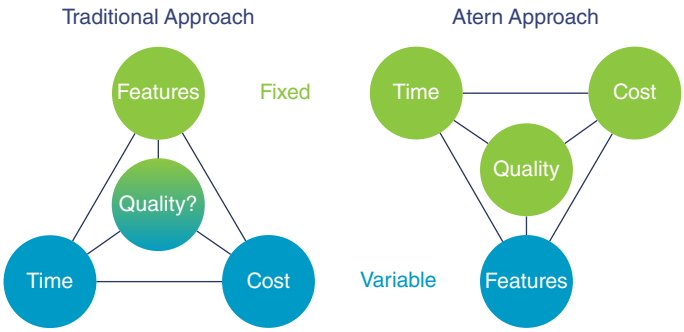


FIGURE 21.1 Project variables DSDM and DSDM Atern

and relationships was helped by the fact that the same people allocated to the projects delivery had previously worked together in preparing the proposal for the opportunity in the first place.

Real-time decision-making: Due to the fixing of time in a DSDM project, real-time decision making was crucial. Key team roles were therefore empowered to make decisions on various tasks and subjects.

Project negotiations: Negotiations were necessary during the project from various perspectives, for example at a technical level with requirement/scope trading. Empowerment was crucial in this regard.

DSDM Overview

DSDM Atern is a hybrid agile approach that is used primarily outside the United States. Figure 21.1 shows the key differences between DSDM Atern and a traditional waterfall approach:

> In the traditional approach to project management (left-hand diagram), the feature content of the solution is fixed while time and cost are subject to variation.
>
> If the project goes off track, more resources are often added or the delivery date is extended. However, adding resources to a late project just makes it later. A missed deadline can be disastrous from a business perspective and could easily damage credibility. Quality is often a casualty and also becomes a variable, accompanied by late delivery and increased cost.
>
> Atern's approach to project management (right-hand diagram) fixes time, cost, and quality at the Foundations phase while contingency is managed by varying the features to be delivered. As and when contingency is required, lower priority features are dropped or deferred with the agreement of all stakeholders in accordance with MoSCoW rules.[6]

(See following discussion for an explanation of MoSCoW.)

[6]DSDM Atern Handbook, http://www.dsdm.org/atern-handbook/flash.html#/15/

DSDM Principles

The following are the key principles behind DSDM:[7]

1. Focus on the business need.
   - Understand the true business priorities.
   - Establish a sound business case.
   - Seek continuous business sponsorship and commitment.
   - Guarantee the minimum usable subset (of features).
2. Deliver on time.
   - Time-box the work.
   - Focus on business priorities.
   - Always hit deadlines.
3. Collaborate.
   - Involve the right stakeholders, at the right time, throughout the project.
   - Ensure that members of the team are empowered to make decisions on behalf of those they represent.
   - Actively involve the business representatives.
   - Build a one-team culture (with the business).
4. Never compromise quality.
   - Set the level of quality at the outset.
   - Ensure that quality does not become a variable.
   - Design, document, and test appropriately.
   - Build in quality by constant review.
   - Test early and continuously.
5. Build incrementally from firm foundations.
   - Strive for early delivery of business benefit where possible.
   - Continually confirm the correct solution is being built.
   - Formally reassess priorities and ongoing project viability with each delivered increment.
6. Develop Iteratively
   - Do enough design up front to create strong foundations.
   - Take an iterative approach to building products.

[7] Ibid.

- Build customer feedback into each iteration.
- Accept that most detail emerges later rather than sooner.
- Embrace change—the right solution will not emerge without it.
- Be creative, experiment, learn, evolve.

7. Communicate continuously and clearly.
   - Run daily team standup sessions.
   - Facilitate workshops.
   - Use rich communications techniques such as modeling and prototyping.
   - Present instances of the evolving solution early and often.
   - Keep documentation lean and timely.
   - Manage stakeholder expectations throughout the project.
   - Encourage informal, face-to-face communications at all levels.
8. Demonstrate control.
   - Use an appropriate level of formality for tracking and reporting.
   - Make plans and progress visible to all.
   - Measure progress through focus on delivery of products rather than completed activities.
   - Manage proactively.
   - Evaluate continuing project viability based on the business objectives.

CHALLENGES

Cultural and Organizational Challenges

Table 21.1 is a summary of the key cultural and organizational challenges faced by General Dynamics, UK and how they were handled.

Contractual Challenges

Table 21.2 is a summary of the key contractual challenges faced by General Dynamics, UK and how they were handled.

Technical Challenges

Table 21.3 is a summary of the technical challenges faced by General Dynamics, UK and how they were handled.

TABLE 21.1 Solutions to Cultural and Organizational Challenges

Challenge	Solution
Creating an Environment of Trust The customer/industry and industry team needed to be transparent while respecting each of the industry partners was a separate commercial organization. The Ministry of Defense (MoD) needed to be comfortable that the government was not being short-changed and that the industry team was making the best possible use of the funding available.	The team, in spite of residing in different parts of the south of England, worked very closely together, either through face-to-face meetings or via telephone conferencing. A lot of the interaction was at the technical level, as that underpinned the project. However, higher-level project management interaction was also required so that the customer was assured the project was achieving its objectives.

TABLE 21.2 Solutions to Contractual Challenges

Challenge	Solution
Using the Contract as a Framework While having a contract between parties was necessary, it needed to allow for a flexible working relationship and ensure all organizations remained committed to common project objections.	The contracting arrangement was such that all parties worked to the same contract milestones. These milestones were linked to payment. The net effect was that all parties were motivated by the project's achievements and success rather than the delivery of individual products and services into the project. The approach aimed to foster a "one for all, all for one" team ethic.
Business Expectations—Risk but No Contingency The DSDM concept of trading scope as a contingency for cost/schedule impacts is alien, but once it was witnessed, General Dynamics bought into the process. The management of cost and schedule through clearly defined time boxes was crucial to minimize risks coming to fruition and hence the need to trade scope.	The metrics produced to demonstrate project performance to the customer similarly helped to reassure General Dynamics, UK's management the project would deliver its objectives within budget. In addition, the monthly financial reviews conducted on all projects within General Dynamics, UK provide further confidence in the project delivery.
Keeping to the Fundamental Rules Preserving fixed cost and schedule while delivering a product at the commensurate quality was a challenge. It was important to recognize that using DSDM and having the opportunity to trade out scope could not be used as an excuse for delivery minimal capability and/or poor quality.	■ The foundation phase of the project was all about defining an achievable scope of work within the time and cost constraints to an acceptable quality. ■ The quality was underpinned by the various engineering development standards produced during the foundation phase.

TABLE 21.3 Solutions to Cultural and Organizational Challenges

Challenge	Solution
Delivering the Required Scope/Capability Ensuring that the ability to trade out requirements was not used as a cop-out for not delivering capability was driven by a robust project management approach.	The delivered project scope/capability and product quality are related in our domain, as the quality is driven by the robustness of the engineering development process. A more robust development process will reduce the scope/capability of the delivered project.
Deriving the Technical Requirements The capability delivered built on existing technology from three independent organizations. The project required further development of the existing technology, as well as new technology across the organizations.	The formation of an integrated project team was key. Although remotely located for significant amounts of time, the use of technology and ensuring proactive behaviors across the team for effective and timely communication mitigated a significant technical and project risk. Organizational boundaries were removed by the individuals involved to ensure, as far as was possible, a seamless, integrated project team.

KEY SUCCESS FACTORS

1. *Relationships, professionalism, and transparency*. The Ministry of Defense (MoD) is a key customer for all the industry partners in their own right. To not recognize the key factors could be potentially damaging to the reputation of the industry partners and have a direct impact on future business. It was important for General Dynamics, UK, as the lead partner, to create the project environment to develop the relationships and ensure professionalism and transparency.
2. *Coaching and mentoring*. The DSDM project delivery technique was new to everyone involved in the project and meant that everyone needed to learn the process together.

   Throughout the project, it was important to ensure that all involved maintained the same common view of the DSDM process that was used for managing the CIdS project. Through team mentoring, a common understanding was established.
3. *Teamwork*. Successful delivery of the CIdS project was achieved through the behaviors and motivations of the people involved—the teamwork.

   - A new project delivery method required key personnel with a pragmatic, flexible, and open-minded approach, coupled with a tolerance for ambiguity and excellent team-building skills.
   - The teamwork has come about through the desire to collaborate, learn together, communicate, and build the necessary relationships to successfully deliver the CIdS project.

- Any stresses in the team dynamic were recognized and dealt with early, before they become issues.

4. *Conflict management*. Through the highly collaborative and open-team approach, with a no-surprises culture, conflict has been kept to a minimum.

   As in most projects there have been stress points during the project, but because of the team relationships, these have been anticipated and quickly resolved. In traditional projects, with their fixation on the achievement of all technical requirements, a defensive posture can often arise if problems in achieving those requirements emerge. The CIdS project, through the highly collaborative approach and safety net of requirements contingency, resulted in a far more open environment where good and bad issues were openly discussed.
5. *Risk management*. The CIdS project benefited from a risk management strategy that used an incremental development approach in which high-risk requirements were tackled early, with requirements flexibility protecting on cost, on schedule, to quality delivery.

   A joint risk register was established to inform time-box management and ongoing engineering development, but not to identify any required financial contingency. Instead of holding financial contingency, any additional funds that were required to ensure delivery of *must have* requirements were drawn from the budget associated with delivery of lesser priority tradable requirements.

CONCLUSIONS

1. *Collaborative approach to contract management*. Through using the DSDM framework on the project, the traditional customer/supplier and prime contractor/subcontractor divides have been bridged to form what truly has been an integrated project team. The one-for-all, all-for-one ethos has prevailed throughout, resulting in a team that focused on delivering on the project objectives rather than what is best for individual organizations.

   Although the DSDM framework has been central to the delivery of the project, it is not a panacea. The CIdS project has ultimately been successful because of the team—their professionalism, their technical capabilities, and their commitment to the principles of DSDM.

> The use of the Dynamic Systems Development Method has been key to the success of the Combat Identification Server TDP: Agile project management techniques, the close involvement of stakeholders throughout, including interim demonstrations, and a constant focus on the deliverables have ensured that the final product will truly hit the mark. This methodology has clearly worked extremely well and I would hope to see lessons from this project applied to future projects.[8]

[8]Major Fiona Galbraith, MoD Sponsor.

2. *More effective project management.* Key benefits that resulted from the CIdS project management approach include the following:
   - More believable plans, schedules, and budgets, and likelihood of adherence were a fundamental outcome as a result of the DSDM approach with its fixed cost and time approach.
   - Using DSDM led to a contract that eliminated financial contingency, delivering maximum possible capability for the funds available.
   - The time-boxing approach resulted in multiple decision points at the end of each time box and each increment, at which alternative ways forward were compared.
   - Through the *Single Statement of User Need,* a common vision of the project objectives was collaboratively developed.
   - With short time-box spans, risk assessment has been aided by the increased focus that resulted.
   - Greater risk taking was enabled as by principle the project could not go over budget, or slip schedule. This allowed desirable but higher-risk requirements to be included as *should* or *could* requirements without fear of penalty to the project. As a result, greater capability has been delivered to the end user for the same money than would otherwise have been possible.

With government funding under continual pressure, successfully demonstrating incremental capability was also a way of reducing the risk of project cancellation by keeping stakeholders engaged and supportive of the CIdS project.

LESSONS LEARNED

1. *Tailor the agile delivery technique as part of early project planning.* From our experience of using DSDM, the process and principles initially seemed quite daunting. In order to successfully apply the technique, it was important to fully understand the objectives of the project through the foundation phase. Having understood the objectives, it was then important to understand DSDM and how DSDM could be used to iteratively deliver the objectives. The lesson was to not rush into the project delivery, and that the early detailed planning is a fundamental part of enabling project success.
2. *Agile techniques can be applied to new project environments.* DSDM's traditional application is internal change projects, particularly IT-related projects. The team led by General Dynamics, UK demonstrated the technique can be applied in a multi-organizational environment, on a project that involves bespoke engineering development. The lesson was, with an open and pragmatic team approach, agile techniques can be adapted for use outside their traditional applications.

22 Overall Summary

HERE ARE SOME OF THE key overall messages that I believe are most important:

1. The project management profession is at a major transition point.

   The project management profession is beginning to go through rapid and profound changes due to the widespread adoption of agile methodologies. Those changes are likely to dramatically change the role of project managers in many environments, as we have known them and raise the bar for the entire project management profession.

2. Agile project management approach:

   - *Don't get too wrapped up in the methodology and language of agile*. Focus on results and adapt the approach as needed to deliver results. That requires an understanding of the principles at a deeper level in addition to the mechanics of how to get things done.
   - *Fit the approach to the project*. Don't make the mistake of force-fitting a project to some predefined, textbook methodology. The right approach is to go in the other direction and fit the methodology to the nature of the project.
   - *Focus on governance, people, process, and tools*. When projects are in trouble and aren't going well, many people try to push harder to make the process work. A better approach is to focus on the systemic factors that are at the core of making an agile project successful:
     - *Governance*: An effective system of project governance to provide oversight over projects to ensure that they effectively fulfill customer needs and manage the company's business interests.
     - *People*: Having the right people on the project where the resources are well trained in the process, technology, and tools, coupled with an approach based on high-performance teams that are cohesive, cross-functional, empowered, and self-organizing.
     - *Process*: Having the right methodology in place that is well designed and appropriate to the nature of the project and is also well integrated with the customer for the project so

that the customer is fully engaged collaboratively in the process in a spirit of partnership and trust.

- *Technology and tools*: Having the right tools in place to support the process and people is extremely important to maximize the efficiency of the people and process, especially in cases that involve distributed teams that are not collocated and the projects are fast-paced and demanding.

- *Delivering faster is extremely important*. Don't get too bogged down in planning and deliver results incrementally if necessary to show progress quickly. Projects generally should not be longer than six months—anything longer than that should probably be considered as a major initiative and broken down into smaller projects as necessary.
- *Do the right thing*. Focus on delivering business value in addition to managing costs and schedules. It can take courage to do the right thing when there is extensive pressure to meet schedule and budget goals. The actual purpose of all projects is to deliver value—budget and schedule are just constraints.
- *Organizational culture is very important*. An agile approach that is not well integrated with the culture of the company it is part of is not likely to be fully successful. However, changing the company's culture so that it is more conducive to agile is desirable if that's possible. It's often not that simple.
- A company's culture should be shaped around whatever makes sense to drive its primary business, and that might or might not be in alignment with an agile development process.
- Changing the culture of a company is not easy to do and is typically not something that can be done quickly.

3. Becoming an agile project manager may require a shift in thinking.

Some of these shifts in thinking include:

- Emphasis on maximizing value versus control
- Emphasis on empowerment and self-organization
- Limited emphasis on documentation
- Managing flow instead of structure
- Cross-functional leadership approach

4. Becoming an agile project manager is a journey.

This book is only the beginning of that journey. Think of *Shu-ha-ri*:

- *Shu*: In the "Shu" stage, the student learns to do things more-or-less mechanically, "by the book," without significantly deviating from the accepted rules and practices and without improvising any new techniques.

- *Ha*: In the "Ha" stage, the student begins to understand the principles at a deeper level and learns how to improvise and break free from rigidly accepted practices, but it's important to go through the "Shu" stage and gain mastery of the foundational principles before you start improvising. You have to learn the foundational principles before you can improvise. Improvisation without knowledge is just amateurish experimentation.
- *Ri*: Finally, in the "Ri" stage, the student gets to the highest level of mastery and is able to develop his/her own principles and practices as necessary.

Appendices

THE APPENDICES TO THE BOOK contain additional information that is intended to supplement the primary information in the rest of the book.

Appendix A

Additional Reading

This appendix contains a list of recommended additional reading to provide a broader and deeper understanding of some of the material in this book.

Appendix B

Glossary of Terms

This appendix contains a list of definitions and terms used throughout the book.

Appendix C

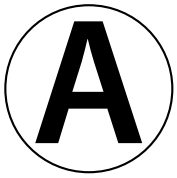
Example Project Charter

This appendix contains an example of a project charter template that can be used in the macro-level of a managed Agile development framework. This charter template is a general model that is intended to be customized as necessary to fit a given project and business environment.

Appendix D

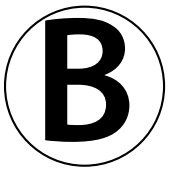
Suggested Course Outline

This appendix contains a suggested course outline for a graduate-level Agile Project Management course based on this textbook.



Additional Reading

Title	Author	Publisher, Date
Agile & Iterative Development—A Manager's Guide	Craig Larman	Addison-Wesley, 2003
Agile Estimating and Planning	Mike Cohn	Prentice Hall, 2006
Agile Project Management	Jim Highsmith	Addison-Wesley, 2010
Agile Project Management with Scrum	Ken Schwaber	Microsoft Press, 2003
Balancing Agility and Discipline—A Guide for the Perplexed	Barry Boehm and Richard Turner	Addison-Wesley, 2003
Effective Project Management Traditional, Agile, Extreme	Robert Wysocki	Wiley, 2014
Implementing Lean Software Development—From Concept to Cash	Mary and Tom Poppendiek	Addison-Wesley, 2007
Leading Lean Software Development	Mary and Tom Poppendiek	Addison-Wesley, 2010
Lean Software Development—An Agile Toolkit	Mary and Tom Poppendiek	Addison-Wesley, 2003
Making Sense of Agile Project Management Balancing Control and Agility	Charles G Cobb	Wiley, 2011
Succeeding with Agile Software Development Using Scrum	Mike Cohn	Addison-Wesley, 2010
The Enterprise and Scrum	Ken Schwaber	Microsoft Press, 2003
The Software Manager's Bridge to Agility	Michele Sliger and Stacia Broderick	Addison-Wesley, 2008



Glossary of Terms

Agile The word *agile* is widely misused to mean many different things in actual practice. For example, in the United States, the term *agile* is often synonymous with the Scrum methodology.

The Merriam Webster Dictionary defines the word *agile* as follows:[1]

- "Marked by ready ability to move with quick easy grace <an agile dancer>"
- "Having a quick resourceful and adaptable character <an agile mind>"

Dictionary.com defines the word *agile* as follows:[2]

- "Quick and well-coordinated in movement; lithe: *an agile leap*."
- "Active; lively: an agile person."
- "Marked by an ability to think quickly; mentally acute or aware: *She's 95 and still very agile*."

Both of those definitions are probably very appropriate in the context of an *agile project manager*:

- A cross-functional agile team moves quickly and autonomously with fluidity and grace without an excessive amount of external direction and orchestration.
- An agile project approach is very adaptive to volatile and uncertain environments as necessary rather than being locked in to a well-defined plan that might be difficult to change.
- Agile people are very resourceful, high-energy people who are able to think quickly in a variety of different circumstances.

Note: You will often see *agile* capitalized.

Agile Modeling Agile Modeling is a methodology developed by Scott Ambler that is a collection of values, principles, and practices for modeling software that can be applied on a software development project in an effective and lightweight manner.[3]

[1] Agile, http://www.merriam-webster.com/dictionary/agile
[2] Agile, http://dictionary.reference.com/browse/agile
[3] Agile Modeling, http://www.agilemodeling.com/

Agile Unified Process (AUP) The Agile Unified Process is a simplified version of the Rational Unified Process (RUP) developed by Scott Ambler. It describes a simple, easy to understand approach to developing business application software using agile techniques and concepts yet still remaining true to the RUP.[4]

Burn-down chart A burn-down chart is a chart often used in scrum agile development to track work completed against time allowed. The *x*-axis is the time frame, and the *y*-axis is the amount of remaining work left that is labeled in story points and man hours, etc. The chart begins with the greatest amount of remaining work, which decreases during the project and slowly burns to nothing.[5]

Burn-up chart A burn-up chart is a graphical representation that tracks progress over time by accumulating functionality as it is completed. The accumulated functionality can be compared to a goal such as a budget or release plan to provide the team and others with feedback. Graphically, the *x*-axis is time and the *y*-axis is accumulated functionality completed over that period of time. The burn-up chart, like its cousin the burn-down chart, provides a simple yet powerful tool to provide visibility into the sprint or program.

The burn-up chart can be thought of as the mirror image of the burn-down chart but is generally extended over multiple sprints to show the strategy being followed as the project builds toward release and product delivery.[6]

Code refactoring Code refactoring involves removing redundancy, eliminating unused functionality, and rejuvenating obsolete designs and improving the design of existing software in order to improve reliability and maintainability of the software. Refactoring throughout the entire project life cycle saves time and increases quality of the software. Code refactoring is more commonly used with Extreme Programming—it is less commonly used with Feature-Driven Development.

Continuous integration Continuous integration is the practice of frequently integrating new or changed software with the code repository. It is a way of early detection of problems that may occur when individual software developers are working on code changes that may potentially conflict with each other. In many typical software development environments, integration may not be performed until the application is ready for final release.

Daily standup A daily stand-up meeting is a short organizational meeting that is held each day in a typical Agile/Scrum project. The meeting, generally limited to between five and fifteen minutes long, is sometimes referred to as a stand-up, daily standup, or a daily scrum.

The purpose of the meeting is for each team member to answer the following three questions:

1. What did you do yesterday?
2. What will you do today?
3. Are there any impediments in your way?

[4]Agile Unified Process, http://www.ambysoft.com/unifiedprocess/agileUP.html
[5]Techopedia—What is a Burn-down Chart?, http://www.techopedia.com/definition/26294/burndown-chart
[6]Cagley, Thomas F., Metrics Minute—Burn Up Charts, http://tcagley.wordpress.com/2011/05/09/metrics-minute-burn-up-charts/

The participants typically stand rather than sit during the meeting to reinforce the idea that the meeting is intended to be short and to discourage wasted time. The stand-up is not meant to be a place to solve problems, but rather to make the team aware of current status. If discussion is needed, a longer meeting with appropriate parties can be arranged separately.[7]

Delphi method A forecasting method based on the results of questionnaires sent to a panel of experts. Several rounds of questionnaires are sent out, and the anonymous responses are aggregated and shared with the group after each round. The experts are allowed to adjust their answers in subsequent rounds. Because multiple rounds of questions are asked and because each member of the panel is told what the group thinks as a whole, the Delphi Method seeks to reach the "correct" response through consensus.[8]

Dynamic Systems Development Model (DSDM) Dynamic Systems Development Method (DSDM) is a framework based originally on Rapid Application Development (RAD), supported by continuous user involvement in an iterative development and incremental approach, which is responsive to changing requirements, in order to develop a system that meets the business needs on time and on budget. DSDM was developed in the United Kingdom in the 1990s by a consortium of vendors and experts in the field of Information System (IS) development, the DSDM Consortium, combining their best-practice experiences.[9] It is most frequently used outside of the United States.

Epic An *epic* is a large user story that needs to be broken down into smaller user stories prior to the start of an agile iteration.

Extreme Programming (XP) Extreme Programming is a discipline of software development based on values of simplicity, communication, feedback, and courage. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.

In Extreme Programming, every contributor to the project is an integral part of the whole team. The team forms around a business representative called "the Customer," who sits with the team and works with them daily.

Extreme Programming teams use a simple form of planning and tracking to decide what should be done next and to predict when the project will be done. Focused on business value, the team produces the software in a series of small fully integrated (see section for more detail).

Feature-driven development Feature-driven development (FDD) is a model-driven approach that puts more emphasis on defining an overall model of the system and a list of features to be included in the system prior to starting the design effort.

Gantt chart A Gantt chart is a type of bar-chart that shows both the scheduled and completed work over a period. A time-scale is given on the chart's horizontal axis and each activity is shown as a separate horizontal rectangle (bar) whose length is proportional to the time required (or taken) for

[7]What is a Daily Standup Meeting?, http://searchsoftwarequality.techtarget.com/definition/daily-stand-up-meeting
[8]Delphi Method Definition, http://www.investopedia.com/terms/d/delphi-method.asp
[9]"What Is DSDM?," Select Business Solutions, http://www.selectbs.com/adt/process-maturity/what-is-dsdm

the activity's completion. In project planning, these charts show start and finish dates, critical and non-critical activities, slack time, and predecessor-successor relationships.[10]

IT strategy IT strategy is a comprehensive plan that information technology management professionals use to guide their organizations.

An IT strategy should cover all facets of technology management, including cost management, human capital management, hardware and software management, vendor management, risk management and all other considerations in the enterprise IT environment. Executing an IT strategy requires strong IT leadership; the chief information officer (CIO) and chief technology officer (CTO) need to work closely with business, budget and legal departments as well as with other user groups within the organization.

Many organizations choose to formalize their information technology strategy in a written document or balanced scorecard strategy map. The plan and its documentation should be flexible enough to change in response to new organizational circumstances and business priorities, budgetary constraints, available skill sets and core competencies, new technologies and a growing understanding of user needs and business objectives.[11]

Kanban board A Kanban board is a columnar status board where each of the columns represents stages in a progression of stages in the flow of a process and items are shown in the column of the board corresponding to their current status in the process flow. A Kanban board could be implemented in the form of a physical board such as a white board with 3X5 cards or "yellow stickies" representing the items in the process or it could be implemented in an online tool.

Lean Manufacturing "Lean manufacturing is a comprehensive term referring to manufacturing methodologies based on maximizing value and minimizing waste in the manufacturing process. Lean manufacturing has evolved in North America from its beginnings in the Toyota Production System (TPS) in Japan. Many of the most recognizable phrases, including kaizen and kanban, are Japanese terms that have become standard terms in lean manufacturing.

At the heart of lean is the determination of value. Value is defined as an item or feature for which a customer is willing to pay. All other aspects of the manufacturing process are deemed waste. Lean manufacturing is used as a tool to focus resources and energies on producing the value-added features while identifying and eliminating non value added activities."[12]

Lean Software Development Lean Software Development is a translation of Lean Manufacturing and Lean IT principles and practices to the software development domain. It is heavily based on the work of Tom and Mary Poppendiek.

Pair Programming Pair programming is an agile software development technique in which two programmers work together at one workstation—one developer plays the role of an observer while the other developer in the pair writes the code.

[10] Gantt Chart Definition, http://www.businessdictionary.com/definition/Gantt-chart.html#ixzz3ArL59bIy
[11] What is IT Strategy?, http://searchcio.techtarget.com/definition/IT-strategy-information-technology-strategy
[12] "Lean Manufacturing," Learning Center, http://www.vorne.com/learning-center/lean-manufacturing.htm

Pert chart A Pert chart is a project management tool that provides a graphical representation of a project's timeline. PERT, or Program Evaluation Review Technique, was developed by the United States Navy for the Polaris submarine missile program in the 1950s. PERT charts allow the tasks in a particular project to be analyzed, with particular attention to the time required to complete each task, and the minimum time required to finish the entire project.
A PERT chart is a graph that represents all of the tasks necessary to a project's completion, and the order in which they must be completed along with the corresponding time requirements. Certain tasks are dependent on serial tasks, which must be completed in a certain sequence. Tasks that are not dependent on the completion of other tasks are called parallel or concurrent tasks and can generally be worked on simultaneously. [13]

Portfolio kanban A "portfolio kanban" is a kanban board that is used as a tool in a portfolio management process to represent the various projects and initiatives that are being managed through each stage of the portfolio management process.

Product backlog The product backlog is a high-level document list of all required features and wish-list items prioritized by business value in an agile project. It is the "what" that will be built. It is owned by the product owner and continuously prioritized and reprioritized as the project progresses, work is completed, and detailed requirements are better understood.
A prototype model is a type of software development lifecycle model that is used to progressively define the requirements for a product or application. It involves developing the requirements as much as possible from user input, then a prototype model is built based on the known requirements. User feedback is then used to progressively refine the model as necessary until it satisfies the user. Prototyping can also be an effective method to demonstrate the feasibility of a certain approach.

Prototype Model The basic reason for limited use of prototyping is the cost involved in this build-it-twice approach. However, with modern software development tools, prototyping need not be very costly and can actually reduce the overall development cost.[14]

QA testing QA testing or software testing is a set of processes aimed at investigating, evaluating and ascertaining the completeness and quality of computer software. Software testing ensures the compliance of a software product in relation with regulatory, business, technical, functional and user requirements. The objectives of these processes can include:[15]

- Verifying software completeness in regards to functional/business requirements
- Identifying technical bugs/errors and ensuring the software is error-free
- Assessing usability, performance, security, localization, compatibility and installation

Rational Unified Process (RUP) The Rational Unified Process (RUP) is a software development process from Rational, a division of IBM. It divides the development process into four distinct

[13] Pert Chart Definition, http://www.investopedia.com/terms/p/pert-chart.asp
[14] Prototyping Software Lifecycle Model, http://www.freetutes.com/systemanalysis/sa2-prototyping-model.html
[15] What is Software Testing?, http://www.techopedia.com/definition/17681/software-testing

phases that each involve business modeling, analysis and design, implementation, testing, and deployment. The four phases are:

Inception: The idea for the project is stated. The development team determines if the project is worth pursuing and what resources will be needed.

Elaboration: The project's architecture and required resources are further evaluated. Developers consider possible applications of the software and costs associated with the development.

Construction: The project is developed and completed. The software is designed, written, and tested.

Transition: The software is released to the public. Final adjustments or updates are made based on feedback from end users. [16]

Scrum Scrum is an agile software development model based on multiple small teams working in an intensive and interdependent manner. The term is named for the *scrum* (or *scrummage*) formation in rugby, which is used to restart the game after an event that causes play to stop, such as an infringement.
Scrum employs real-time decision-making processes based on actual events and information. This requires well-trained and specialized teams capable of self-management, communication and decision-making. The teams in the organization work together while constantly focusing on their common interests.[17]

Software Development Lifecycle (SDLC) A software development lifecycle (SDLC) is a process or framework that describes the activities performed at each stage of the software development process and provides an overall roadmap for the project. It provides a template for executing a project that can be tailored to fit a particular project. An example of an SDLC would be the waterfall model; however, as used in this book, an SDLC would include any project framework, such as Scrum.

Spike "A spike is an experiment that allows developers to learn just enough about the unknown elements in a user story, e.g. a new technology, to be able to estimate that user story. Often, a spike is a quick and dirty implementation or a prototype, which will be thrown away. When a user story on the product backlog contains unknown elements that seriously hamper a usable estimation, the item should be split into a spike to investigate these elements plus a user story to develop the functionality."[18]

Sprint In product development, a sprint is a set period of time during which specific work has to be completed and made ready for review.

Story Points Story points are a method of estimating the level of effort associated with implementing a user story. The typical form of story points is a Fibonacci series such as 1, 2, 3, 5, 8, 13,

[16] RUP (Rational Unified Process Definition), http://www.techterms.com/definition/rup
[17] "What Is Scrum?," http://searchsoftwarequality.techtarget.com/sDefinition/0,,sid92_gci1230820,00.html.
[18] Erik Phillipus, "Architecture Spikes," http://www.agile-architecting.com/Architecture%20Spikes.pdf.

with 1 being a minimal level of effort and 13 being a maximum level of effort for a user story in an iteration.

Test-Driven Development (TDD) Test-Driven Development (TDD) is commonly used in agile methodologies to integrate testing directly into the software development effort:
"Instead of writing functional code first and then your testing code as an afterthought, if you write it at all, you instead write your test code before your functional code. Furthermore, you do so in very small steps—one test and a small bit of corresponding functional code at a time. A programmer taking a TDD approach refuses to write a new function until there is first a test that fails because that function isn't present. In fact, they refuse to add even a single line of code until a test exists for it. Once the test is in place they then do the work required to ensure that the test suite now passes."[19]

Total Quality Management (TQM) Total Quality Management (TQM) is a holistic approach to long-term success that views continuous improvement in all aspects of an organization as a process and not as a short-term goal. It aims to radically transform the organization through progressive changes in the attitudes, practices, structures, and systems.
Total quality management transcends the product quality approach, involves everyone in the organization, and encompasses its every function: administration, communications, distribution, manufacturing, marketing, planning, training, etc. Coined by the US Naval Air Systems Command in early 1980s, this term has now taken on several meanings and includes:

- Commitment and direct involvement of highest-level executives in setting quality goals and policies, allocation of resources, and monitoring of results;
- Realization that transforming an organization means fundamental changes in basic beliefs and practices and that this transformation is everyone's job;
- Building quality into products and practices right from the beginning;
- Understanding of the changing needs of the internal and external customers, and stakeholders, and satisfying them in a cost effective manner;
- Instituting leadership in place of mere supervision so that every individual performs in the best possible manner to improve quality and productivity, thereby continually reducing total cost;
- Eliminating barriers between people and departments so that they work as teams to achieve common objectives; and
- Instituting flexible programs for training and education, and providing meaningful measures of performance that guide the self-improvement efforts of everyone involved.[20]

[19] "Introduction to Test-Driven Design," http://www.agiledata.org/essays/tdd.html
[20] What is Total Quality Management?, http://www.businessdictionary.com/definition/total-quality-management-TQM.html

User Persona A *user persona* is a description of a specific type of user that impacts the requirements. For example, a *banking customer* would be an example of a user persona. User personas are useful ways of characterizing the users of the system and keeping the development effort focused on satisfying their needs.

User Stories User stories are one of the primary development artifacts for Scrum and Extreme Programming (XP) project teams. A user story is a high-level definition of a requirement, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement it.

User stories are much smaller than other usage requirement artifacts such as use cases or usage scenarios. User stories provide a way of breaking up the project into individual work items that can provide a way of estimating and tracking work to be done. Each user story may be further refined as necessary as the design progresses.

Velocity "Velocity is the measure of the throughput of an agile team per iteration. Since a user story, or story, represents something of value to the customer, velocity is actually the rate at which a team delivers value. More specifically, it is the number of estimated units (typically in story points) a team delivers in an iteration of a given length and in accordance with a given definition of 'done.'"[21]

Waterfall Model The waterfall model is a software development lifecycle model that describes a linear and sequential development method. Waterfall development has distinct goals for each phase of development and once a phase of development is completed, the development typically proceeds to the next phase and there is no turning back.

The advantage of waterfall development is that it allows for departmentalization and managerial control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process sequentially. Development moves through the phases of the model such as concept, design, implementation, and testing, and ends up at deployment and operation and maintenance. Each phase of development normally proceeds in a prescribed order, without any overlapping or iterative steps; however, tailoring is often done to make the process more efficient.

The disadvantages of waterfall development are that:

- It does not allow for much reflection or revision—once an application is in the testing stage, it is very difficult to go back and change something that was not well thought out in the concept stage.
- It assumes that all requirements can be defined upfront prior to any design work and that is not a realistic assumption in many situations and it doesn't provide an approach that is flexible and adaptive to customer needs.

It may require an excessive amount of documentation artifacts.[22]

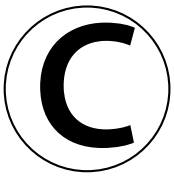
[21] "Agile Sherpa—Velocity," http://www.agilesherpa.org/agile_coach/metrics/velocity/

[22] "What Is Waterfall Model," http://searchsoftwarequality.techtarget.com/sDefinition/0,,sid92_gci519580,00.html.

Work Breakdown Structure (WBS) A Work Breakdown Structure (WBS) is a project network-modeling step in which the entire project is graphically subdivided into manageable work elements (tasks) that is typically shown in a hierarchical structure. WBS displays the relationship of each task to the other tasks, to the whole and the end product (goal or objective).[23]

XP See Extreme Programming.

[23]What is a Work Breakdown Structure?, http://www.businessdictionary.com/definition/work-breakdown-structure-WBS.html#ixzz3ArUoGY3m



Example Project/ Program Charter Template

THIS APPENDIX CONTAINS AN EXAMPLE of a project charter template that can be used to define the macro layer in a hybrid, managed agile development approach. This template is provided as an example and is intended to be customized to fit the project and business environment that it is used in.

Project overview

Background

Provide a brief description of the background behind the problem that the project or program is intended to address to a sufficient level to allow the reader to understand the context of the problem.

Problem Statement

Provide a brief description of the problem that the project or program is intended to address from a business or operational management perspective.

Project Vision

Write a concise vision statement that summarizes the purpose and intent of the project and describes what the world will be like when the project is completed. The vision statement should reflect a balanced view that will satisfy the needs of diverse customers as well as those of the developing organization. It may be somewhat idealistic, but it should be grounded in the realities of existing or anticipated

customer markets, enterprise architectures, organizational strategic directions, and cost and resource limitations. Consider using the following template:

- For (target customer)
- Who (statement of the need or opportunity)
- The (product name)
- Is a (product category)
- That (key benefit, compelling reason to buy or use)

Success Criteria

What are the success criteria for the project? How do you know if the project has been successful?

Project Approach/Development Process

Identify the development process and/or any deviations from the standard methodology that will be used for this project or program.

Project plan

This section outlines the plan for managing the project.

Scope

The project scope defines the range of the proposed products and services the project will deliver. Scope can be represented using a context diagram, an event list, and/or a feature table. Scope might be subdivided into the scope of the initial product release and planned growth strategies for subsequent releases. It is also important to define what the project will not include, so describe limitations and exclusions, such as product features or characteristics that a stakeholder might anticipate, but which are not planned to be included in the project.

In Scope

The project scope provides an overview of the user stories that the project will deliver. Scope might be subdivided into the scope of the initial product release and planned growth strategies for subsequent releases.

Release	Priority	Story #	Story Name	Description

Out of Scope

It's also important to define what the project will not include, so describe limitations and exclusions, such as product features or characteristics that a stakeholder might anticipate, but which are not planned to be included in the project.

- Out of Scope Item #1
- Out of Scope Item #2
- Etc.

Related Projects and Systems

Identify any related projects or systems and describe the interrelationship.

Project or System	Interrelationship

Project Participants

Identify key project participants and the role that they will play in the project.

Role	Name	Organization
Business Sponsor		
Product Owner		
Business Process Owner		
Subject Matter Experts		
Stakeholders		
Project Manager		
Business Analyst		

Constraints, assumptions, and risks

Constraints

Constraints are impediments and other factors that must be considered when executing the project or program. The constraints that are applicable to this program include:

Identify known project constraints, such as products to be reused, components to be acquired, interfaces to other projects or products, or technologies to be employed.

Constraint	Impact

Assumptions

Record any assumptions that were made (as opposed to known facts) when conceiving the project. An assumption is a statement that CANNOT be proven to be true, but needs to be taken as being true in order to proceed with the solution. Since the assumption is essential to progress, it is also a dependency that carries some risk because of its indeterminate nature, and the associated risk should be noted in the Business Risks section below.

- Assumption #1
- Assumption #2
- Etc.

Dependencies

Note any major external dependencies the project must rely upon for success, such as specific technologies, third-party vendors, development partners, or other business relationships. Also, identify any other projects that are related to this project in some way or may have a bearing on its outcome.

- Dependency #1
- Dependency #2
- Etc.

Risks

Summarize the major business risks associated with this project or program, such as marketplace competition, timing issues, user acceptance, implementation issues, or possible negative impacts on the business. Estimate the severity of each risk's potential impact and identify any risk mitigation actions that could be taken. This is not the place for the project's overall risk list.

Risk	Mitigation Strategy

Timeline Estimate

The following table contains an estimated schedule for the major milestones in the project.

Include a list of major project milestones and key deliverables with estimated target dates.

Date	Milestone	Deliverables

D Suggested Course Outline

THIS BOOK IS DESIGNED TO be used to support a graduate-level course in Agile Project Management. The following is a suggested course outline that can be used with the course.

Course Overview

The course provides an understanding of how new Agile principles and practices are changing the landscape of project management and is designed to give experienced project managers fresh new insight into how to successfully blend Agile and traditional project management principles and practices in the right proportions to fit any business and project situation. It is desirable but not essential for the students to have a project management background as well as a basic foundation of knowledge of agile principles and practices. The course provides a much deeper understanding of both Agile and traditional project management principles and practices in order to see them as complementary rather than competitive approaches. Topics include:

- Agile fundamentals, principles, and practices including the Scrum methodology
- Understanding Agile at a deeper level and the roots of Agile
- Agile Project Management Principles and Practices, Tools and Techniques
- Adapting an Agile approach to fit a business environment at an enterprise level
- Planning and managing an enterprise-level Agile transformation for a business
- Scaling agile to an enterprise level (including the use of enterprise-level Agile frameworks and Agile Project Management tools)
- Enterprise-level Agile Transformations
- The course uses case studies and interactive discussions extensively to provide students with a real-world perspective of how to put these concepts into action.

Course Objectives:

The general objective of the course can be summed up as giving experienced project managers the tools and skills to assume a major leadership role in applying a new and emerging Agile Project Management approach to large and complex enterprise-level initiatives and to develop a more adaptive project management approach that can be applied to almost any project. It is my objective to give you a very fresh new outlook on how to blend Agile and more traditional plan-driven Project Management principles and practices to dramatically enhance your repertoire of project management capabilities. In addition, you will learn how to adapt these principles and practices to different business environments and how to lead overall Agile business transformations.

The more specific objectives of the course are:

1. Develop new ways of thinking and begin to see Agile principles and practices in a new light as complementary rather than competitive to traditional project management practices
2. Gain an understanding of the fundamentals of Agile practices and learn the principles behind the Agile practices at a deeper level in order to understand why they make sense and how they can be adapted as necessary to fit a given situation
3. Learn how to go beyond the traditional notion of plan-driven project management and develop an adaptive approach to project management that blends both Agile and traditional project management principles and practices in the right proportions to fit a given project and business environment
4. Understand the potential roles that an Agile Project Manager can play and begin to reshape project management skills around those roles
5. Learn some of the challenges of scaling Agile to an enterprise level and develop experience in applying these concepts in large, complex enterprise-level environments

Course Outline:

Course Outline	Reference	Lesson
Part 1 – Fundamentals of Agile		
1. Introduction, Course Objectives, and Agile Overview		1
1.1. Introductions and Course Objectives		1
1.2. Evolution of the Project Management Profession	Chapter 1	1
1.3. Agile Overview	Chapter 1	1
1.3.1. What is Agile?	Chapter 1	1
1.3.2. Agile Perception vs. Reality	Chapter 1	1
1.3.3. Agile Project Management Benefits	Chapter 1	1
Assignment: Complete Discussion Topics		

(*Continued*)

Course Outline	Reference	Lesson
2. Agile Fundamentals		2
2.1. Agile History, Values, and Principles	Chapter 2	2
Assignment: Complete discussion topics		
3. Scrum Overview	Chapter 3	3
3.1. Scrum Roles	Chapter 3	3
3.2. Scrum Methodology	Chapter 3	3
3.3. General Scrum/Agile Principles	Chapter 3	3
3.4. Scrum Values	Chapter 3	
Assignment: Complete Discussion Topics Draft Proposal for Research Paper Due		
4. Overview of Agile Practices		4
4.1. Agile Planning/Requirements Practices & Product Backlog	Chapter 4	4
4.1.1. General Agile Planning Practices	Chapter 4	4
4.1.2. General Agile Requirements Principles & Practices	Chapter 4	4
4.1.3. Product Backlog	Chapter 4	4
4.2. Agile Development, Quality, & Testing Practices	Chapter 5	4
4.2.1. Agile Development Practices	Chapter 5	4
4.2.2. General Agile Quality Management Practices	Chapter 5	4
4.2.3. Agile Testing Practices	Chapter 5	4
Assignment: Complete Discussion Topics & Product Backlog Assignment		
Part 2 – Agile Project Management		
5. Time Boxing, Kanban, Theory of Constraints, Estimation	Chapter 6	5
5.1. Kanban Process Overview	Chapter 6	5
5.2. Theory of Constraints	Chapter 6	5
5.3. Agile Estimation	Chapter 7	5
Assignment: Complete discussion Topics and Theory of Constraints Assignment Complete Estimation Exercise		
6. Agile Project Management Role		6
6.1. Levels of Agile Implementation	Chapter 8	6
6.2. The Role of an Agile Project Manager	Chapter 8	6
6.2.1. Agile Project Management Shifts in Thinking	Chapter 8	6
6.2.2. Agile and PMBOK	Chapter 8	6
6.2.3. Potential Agile Project Management Roles	Chapter 8	6
Mid-term Exam Final Research Paper Proposal Due		

(*continued*)

(*Continued*)

Course Outline	Reference	Lesson
7. Agile Communications Practices & Tools	Chapter 9	8
7.1. Agile Communications Practices	Chapter 9	8
7.2. Agile Tools	Chapter 9	8
7.3. VersionOne Tool Overview	Chapter 10	8
Assignment: Complete Discussion Topics & Initial VersionOne Lab		
8. Understanding Agile at a Deeper Level		9
8.1. Systems Thinking	Chapter 11	9
8.2. The Influence of Total Quality Management (TQM)	Chapter 11	9
8.3. The Influence of Lean Manufacturing	Chapter 11	9
Assignment: Complete Discussion Topics		
Part 3 – Making Agile Work for a Business		
9. Enterprise-level Agile Implementation		10
9.1. Scaling Agile to an Enterprise Level	Chapter 12	10
9.1.1. Enterprise-level Agile Development Practices	Chapter 12	10
9.1.2. Enterprise-level Management Practices	Chapter 12	10
9.2. Adapting an Agile Approach to Fit a Business	Chapter 13	10
9.2.1. The Impact of Different Business Environments	Chapter 13	10
9.2.2. Corporate Culture and Values	Chapter 13	10
Assignment: Complete Discussion Topics		
10. Enterprise Level Management Systems		11
10.1. Enterprise-level Management Practices	Chapter 13	11
10.2. Adaptive Project Governance Model	Chapter 14	11
10.3. Enterprise-level Transformations and Change Management	Chapter 14	11
Assignment: Complete Discussion Topics		
11. Enterprise-level Agile Frameworks		12
11.1. Scaled Agile Development Framework	Chapter 15	12
11.2. Managed Agile Development Framework	Chapter 16	12
11.3. Disciplined Agile Delivery Framework	Chapter 17	12
Assignment: Complete Discussion Topics		

(*Continued*)

Course Outline	Reference	Lesson
12. Case Studies		13
12.1. Not-so-successful Case Studies	Chapter 18	13
12.2. Valpak	Chapter 19	13
12.3. Harvard Pilgrim Healthcare	Chapter 20	13
12.4. General Dynamics, UK	Chapter 21	13
Assignment: Complete Discussion Topics & Final Research Paper Due		
13. Presentation of Student Research Papers		14
Final Exam		15

INDEX

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.