

Compliments of  blueprint

Limited Edition

Enterprise Agile Requirements

FOR
DUMMIES[®]
A Wiley Brand

Learn:

- The basics of agile
- Key concepts of enterprise agile
- How user stories relate to requirements
- How to choose the right solution for enterprise agile requirements

Doug Stewart
Tony Higgins



About Blueprint, the Requirements Company

Blueprint develops best-in-class requirements definition and management (RDM) software. By solving many of the time-consuming, costly, and error-prone elements of defining requirements, Blueprint helps companies ensure that their complex and mission critical software and IT projects are completed on time and budget.

For more information, go to www.blueprintsys.com/.

Enterprise Agile Requirements

FOR
DUMMIES[®]
A Wiley Brand

Limited Edition

**by Doug Stewart
and
Tony Higgins**

FOR
DUMMIES[®]
A Wiley Brand

Enterprise Agile Requirements For Dummies®, Limited Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

Copyright © 2014 by John Wiley & Sons, Inc., Hoboken, New Jersey

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

Trademarks: Wiley, the Wiley logo, For Dummies, the Dummies Man logo, A Reference for the Rest of Us!, The Dummies Way, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and other countries, and may not be used without written permission. Blueprint and the Blueprint logo are trademarks or registered trademarks of Blueprint Software Systems, Inc. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc., is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, or how to create a custom *For Dummies* book for your business or organization, please contact our Business Development Department in the U.S. at 877-409-4177, contact info@dummies.biz, or visit www.wiley.com/go/custompub. For information about licensing the *For Dummies* brand for products or services, contact BrandedRights&Licenses@Wiley.com.

ISBN 978-1-118-84937-8 (pbk); ISBN 978-1-118-85000-8 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Development Editor: Brian Underdahl
Project Editor: Beth Taylor
Acquisitions Editor: Connie Santisteban
Editorial Manager: Rev Mengle

Business Development Representative:
Christiane Cormier
Custom Publishing Project Specialist:
Michael Sullivan
Project Coordinator: Melissa Cossell

About the Authors

Tony Higgins is a vice president at Blueprint and a leading expert on all things software application lifecycle related. Tony has amassed a broad base of skills and experience in software and technology marketing, development, delivery and enablement.

Having worked with both start-up and enterprise-level organizations over a span of more than 28 years, Tony offers a comprehensive perspective on both the technical and business requirements that drive successful implementation results.

Over the course of his career, Tony has worked on numerous military projects involving battlefield command, underwater acoustics, and supply systems. He has also worked on several commercial projects, including broadcast/media applications in addition to business IT projects.

Known for his lead-by-example approach, Tony has shared his technical and marketing leadership knowledge at numerous high profile industry events in North America and Europe, including HP Software Universe, Business Analyst World and Project World and World Congress for Business Analysts. He has educated audiences in such areas as requirements definition and management, business analysis, and systems and software modeling.

Doug Stewart is no stranger to software development and Agile solution delivery. As a director of professional services for Blueprint Software Systems, Doug brings with him more than 30 years of experience in application development, where he has played a wide variety of roles, from business and system analysis to deployment, tech sales support, and project management. He is a key architect behind the Blueprint for Enterprise Agile strategy and product offering.

A veteran trainer and instructor, Doug is a keen proponent of demystifying Agile development for customers, while ensuring they meet corporate and regulatory standards. He does this by drawing upon his extensive experience in development and prototyping techniques.

His work includes a diverse range of large-scale projects, including municipal GIS systems, social networking, accounting, and insurance IT applications. He has also worked extensively on aerospace and military software development projects.

His particular subject areas of note are enterprise agile requirements and project management, as well as process improvement and change agency.

Table of Contents

Introduction.....	1
About This Book	1
Icons Used in This Book.....	1
Where to Go from Here	2
Chapter 1: Introducing Agile	3
Looking at Agile Basics.....	3
Agile's history	3
Agile's values.....	5
The iron triangle	6
Understanding Agile Principles.....	7
Getting to Know Enterprise Agile.....	7
Project-oriented agile methods.....	7
Scaling agile for the enterprise	8
Chapter 2: Addressing Issues with Scaling Agile for IT	11
Seeing the Current State	11
Understanding the Clash of Business and Development....	13
Looking at Current Agile Practice.....	14
User stories are transient	15
Little organized abstraction.....	16
No auditable tracing.....	16
No analyzing of the business.....	16
No good solution for managing complex needs	17
No ongoing record.....	17
A belief that requirements are not needed	17
Chapter 3: Understanding the Role of Requirements ..	19
Understanding the Different Audiences.....	19
Achieving Compliance through Practical Governance	21
Supporting Reuse.....	23
Reuse for savings and efficiency.....	24
Reuse for standardization and consistency.....	24
Reducing effort and risk.....	24
Managing Change.....	25
Seeing an abstract view of change impact	25
Estimating the effort of making a change	26
Making better business decisions	26

Chapter 4: Gaining Advantage with Enterprise Agile Requirements	27
Defining Roles	27
Product owner	27
Business analyst	28
Understanding Requirements in Enterprise Agile	29
Understanding Why User Stories Aren't Your Requirements	31
Executing Your Project	32
Defining business requirements to win the business case	32
Define system requirements during sprint execution	33
Layer-up persistent requirements definition while burning down the product backlog	34
Just enough detail, just in time	35
Chapter 5: Using Blueprint for Enterprise Agile	37
Getting to Know Blueprint	37
Applying Blueprint for Enterprise Agile	39
Looking at a Sample Project	40
Chapter 6: Ten Tips for Enterprise Agile Requirements	47
Avoid Depending on User Stories Alone	47
Analyze the Business	48
Use Traceability—But Just Enough	48
Lean on a Framework	48
Stagger Your Horizons	49
Get Visual	49
Define Your Roles	50
Do Just Enough Up Front	51
Make Other Groups First-Class Citizens	51
Use Modern Technology	52

Introduction

.....

Welcome to *Enterprise Agile Requirements For Dummies*, Limited Edition. Organizations are finding that they need better ways to develop the applications that are critical to the enterprise. The existing old-fashioned methods are too slow and cumbersome to meet the competitive demands of being nimble and responsive. Quite simply, you need to be more agile than traditional methods allow.

About This Book

Enterprise Agile Requirements For Dummies shows you how your organization can adopt an agile approach to developing the critical applications you need to survive. In this book, we introduce you to agile methodology and explain how this new approach differs from traditional methods.

This book also introduces you to Blueprint, an integrated business application designed to help make success with enterprise agile much easier and far more efficient.

Icons Used in This Book

This book uses the following icons to call your attention to information you may find helpful in particular ways.



The information marked by this icon is important and therefore repeated for emphasis. This way, you can easily spot noteworthy information when you refer to the book later.



This icon points out extra-helpful information.



Seek out the On the Web icon if you want to find out even more about enterprise agile requirements.

Where to Go from Here

This book is compact and concise, so we hope you read everything and don't miss something important. But we realize that you may not have time to sit down and read the whole thing right now and maybe you really need a quick start on one particular element of being agile. That's okay, each chapter is designed to stand on its own and provide you with a lot of useful information. But if you do jump right into the middle, we hope you take the time later to read the rest of the book. Otherwise, you will never know what you missed!

Chapter 1

Introducing Agile

In This Chapter

- ▶ Understanding agile for enterprise
- ▶ Looking at the principles
- ▶ Introducing enterprise agile

To get started making your enterprise more agile, you need to know what being agile really means. Sure, you probably know the dictionary definition of the word *agile*, but that dictionary definition doesn't really mention anything about software development in the enterprise, does it? This chapter tackles the basics of agile for enterprise software development.

Looking at Agile Basics

Software development has been going on since the first computers were created many decades ago. Over the years, many different methods for software development have come and gone. Some of those methods were more effective than others. Traditionally, organizations have taken relatively rigid approaches to the software development process.

Agile's history

Traditional software development methods have their roots from a time when computers were a novelty and people didn't know much about programming. These days, it's hard to even imagine a time when people ran large companies without business applications. For that matter, it's hard to even consider how even a small company today can operate without access to such software.

Because virtually every enterprise depends on computers, software development is vital. But even more than that, being able to develop the right software, and to do so quickly, is often key to your company's competitiveness.

The need for better software development methods led a group of software developers to meet in February of 2001. At this meeting, the group discussed a number of principles that defined an approach that became known as *agile software development*.

After the meeting, the group published the principles in the *Manifesto for Agile Software Development*. (See the "Defining the 12 principles of the Agile Manifesto" sidebar for a complete list of the principles.) This document defined the agile software development approach in which they made several declarations, which together they believed described a superior way of developing software.

This set of 12 principles is intended to improve software development by

- ✓ fostering cooperation.
- ✓ reducing documentation as much as possible.
- ✓ working towards incremental goals.
- ✓ trusting people to complete their tasks professionally.

Agile programming methods typically set small goals that can be reached quickly by agile teams, but which often do not result in a finished product until quite a number of iterations have been performed.



You can find the complete manifesto online at <http://agilemanifesto.org/>.



Being agile means developing software according to the Agile Manifesto principles. How you do this depends on your specific situation.

Defining the 12 principles of the Agile Manifesto

As we mention earlier, agile is not a software development process, but rather a set of principles for software development. The following list details the principles:

- ✓ Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- ✓ Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- ✓ Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- ✓ Business people and developers must work together daily throughout the project.
- ✓ Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- ✓ The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- ✓ Working software is the primary measure of progress.
- ✓ Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- ✓ Continuous attention to technical excellence and good design enhances agility.
- ✓ Simplicity—the art of maximizing the amount of work not done—is essential.
- ✓ The best architectures, requirements, and designs emerge from self-organizing teams.
- ✓ At regular intervals, the team reflects on how to become more effective, and then tunes and adjusts its behavior accordingly.

Agile's values

The group of software developers who defined the principles also defined a set of values in which they declared that certain things had more value than others. Specifically, they said that they came to value the following:

- ✓ Working with individuals and interactions over processes and tools
- ✓ Working with software over comprehensive documentation
- ✓ Collaborating with customers over contract negotiation
- ✓ Responding to change over following a plan

Many people have naively interpreted these ideas as absolutes, but the realities of large distributed enterprises, compliance requirements, corporate policies, and procedures have led to the natural evolution of agile concepts for the enterprise, some of which we address in this book.

The iron triangle

Organizations wishing to become agile also need to understand the *iron triangle*—a term commonly used to represent the relationship between the constraints that apply to projects. These constraints include the scope, resources, and schedule for the project, as shown in Figure 1-1. The traditional view locks these three constraints, but a key concept in agile recognizes that the project scope varies while resources and schedule remain constant.

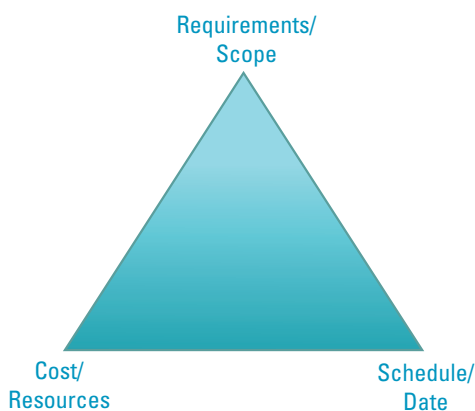


Figure 1-1: The iron triangle.

Understanding Agile Principles

As we mention earlier in this chapter, Agile software development follows a set of 12 basic principles. These principles define agile, but they aren't a process that can be followed specifically. Rather, agile developers adopt some number of the principles that taken together collectively define their process.



Because the principles don't define a process, enterprises can choose which of the principles to adopt and how to adopt them to best suit their situation. Some popular agile methods, such as Scrum and Extreme Programming, are collections of practices that reflect the agile principles in a project-focused way but have mixed results when an attempt is made to scale them to the enterprise.

Agile software development methods typically break tasks into small increments that don't require extensive planning. Rather than working in isolation, developers work as part of cross-functional teams that have members from many different areas of the organization. Development progresses through small increments with very frequent discussions and demonstrations. Using this process allows the project to quickly adapt as it becomes apparent which changes are needed. Priorities are assigned on an as-needed basis and can be quickly changed as development continues. In most cases, little emphasis is placed on documentation, with face-to-face conversation taking the place of written documents.

Getting to Know Enterprise Agile

Now that you've seen some of the basics of agile, take a look at exactly what enterprise agile is.

Project-oriented agile methods

Several popular software development methods promote agile software development. These methods include Scrum, Extreme Programming (XP), and Feature Driven Development (FDD). Briefly, these methods are as follows:

- ✓ **Scrum:** This is an iterative, incremental method for managing projects where development teams work as units to reach a common goal by encouraging co-location of team members and verbal communications.
- ✓ **Extreme programming:** This method is intended to improve software quality and responsiveness to changing customer requirements through frequent and short development cycles.
- ✓ **Feature-driven development:** This method uses milestones to mark the progress through five basic activities. These activities include developing the overall model, building the feature list, planning by features, designing by features, and building those features.

Scaling agile for the enterprise

Very large organizations (or enterprises) certainly have different needs than do small companies.

Very large organizations tend to have very large IT software projects. These projects involve large numbers of people divided into multiple teams who are often distributed geographically. These projects often involve using outsourcers—sometimes several—for various parts of the development and testing work.

Additionally, in very large organizations, a long list of business constraints often must be accommodated by the project. Some of these constraints are self-imposed by corporate policy, while others may come from external bodies in the form of regulations that need compliance.

The software application being built typically has to fit into a larger picture, not only interoperating with many other applications in the business but also needing to adhere to an overall enterprise architecture.

Chapter 2 discusses at length some of the issues that enterprises encounter when trying to scale agile to meet their needs. Some of the things that you can do to foster agile development methods in your organization are listed in the following sections.

Adopting agile

When adopting agile methods in your enterprise, use the following tips:

- ✓ **Make the process your own.** Taking ownership of the agile process is an important first step because it takes leadership to overcome the embedded bias against change.
- ✓ **Build upon the strengths of project-oriented agile.** Finding the places where agile methods can make the most immediate impact helps to ensure that your transition will be smoother.
- ✓ **Evolue the strengths of traditional enterprise methods to agile.** Moving to agile doesn't have to mean abandoning methods that are currently working well for you. In fact, one of the mistakes organizations frequently make when "going agile" is believing that they need to tear it all down and rebuild from the ground up while discarding activities that have been working well. Incorporating what is working well will also aid in the acceptance of change in software development due to the adoption of agile.
- ✓ **Embrace change.** Be prepared to start with some key principles or a predefined process, and allow for process improvement on a regular basis as you find out more about applying agile in your situation.

Adding agile project management principles

In addition to the 12 basic principles mentioned earlier in this chapter, some members of the software development community decided that a set of project management principles were also appropriate. These six project management principles include the following:

- ✓ Increase return on investment by making continuous flow of value our focus.
- ✓ Deliver reliable results by engaging customers in frequent interactions and shared ownership.
- ✓ Expect uncertainty and manage for it through iterations, anticipation, and adaptation.

- ✓ Unleash creativity and innovation by recognizing that individuals are the ultimate source of value and creating an environment where they can make a difference.
- ✓ Boost performance through group accountability for results and shared responsibility for team effectiveness.
- ✓ Improve effectiveness and reliability through situationally specific strategies, processes, and practices.

As with the basic principles, you are free to choose the project management principles that suit your needs and apply them with consideration to your context. Together, the principles define what it means to be agile. Enterprises adopt practices consistent with these principles but may do so differently than organizations focused on project-focused results.



To find out more about RDM in general, you may want to obtain a copy of *Requirements Definition & Management For Dummies*. Visit www.blueprintsys.com/asset/ebook-requirements-definition-and-management-for-dummies/ to obtain a copy of this book.

Chapter 2

Addressing Issues with Scaling Agile for IT

In This Chapter

- ▶ Understanding current realities
 - ▶ Getting past conflicts between developers and business
 - ▶ Seeing what agile practice is today
-

Virtually every enterprise encounters a certain number of issues when transforming to a new process. Adapting to agile software development and reconciling the different needs of IT and the business end of the organization requires addressing a number of important items.

This chapter looks at the current state of enterprise software development and how the needs of business and development can differ.

Seeing the Current State

Today's business environment is highly competitive. Organizations need to use every tool possible in order to be successful in the market. Because every business function relies on business software to some degree, how good these business applications are and how fast they can be put into service directly impacts a company's competitiveness. Most large enterprises are trying to adopt agile practices so they can accelerate software delivery and improve quality in order to compete profitably.

As enterprises attempt to adopt agile practices, they often struggle trying to reconcile traditional essential business functions with an agile approach. These traditional, yet essential, business functions include

✓ **PPM (Project and Portfolio Management):** In the enterprise, there is not one agile project, but collections, or portfolios, of projects:

- Within the portfolio some projects may already be in progress, others not.
- Within the portfolio some projects may be agile, others not.
- All of the projects in the portfolio may share common fundamentals for software developed in the enterprise.

✓ **Scoping:** All software projects need clearly understood boundaries in order to stay focused on developing the right system.

✓ **Business cases:** Business cases describe the desired value of new applications or updates to existing ones in comparison to anticipated costs.

✓ **Funding:** Enterprises have more opportunity for new software applications than available budget for developing them.

✓ **Business analysis:** Enterprises identify business needs and define solutions for those needs.

✓ **Compliance and audit:** Many enterprises have needs to meet legislation or other standards of quality and to be able to prove compliance in an audit.

✓ **Release management:** In an enterprise portfolio of projects, it is often the case that software is made available to users at predetermined times in order to coordinate with other applications' availability or simply to meet expectations in the market.

These essential business functions have traditionally followed a fairly rigid path that has been defined by many levels of advanced planning and documentation. This approach contrasts quite sharply with the new agile approach.

Often new approaches, such as agile practices, are successfully adopted by innovators in the enterprise. People naturally resist any change, so the adoption of agile tends to face resistance when the projects are first rolled out to the early adopters, the early majority, and other users in the enterprise. In the next section, you find out why a conflict often occurs between the interests of software developers and business.

Understanding the Clash of Business and Development

Understanding why traditional business methods can clash with the whole concept of agile software development is important. Although both traditional business methods and agile software development ultimately have the same goal of fostering the enterprise's interests, the two methods have some large fundamental differences.

Traditional business methods are driven by needs that have developed over many years. Extensive advanced planning and documentation are generally considered integral parts of traditional business processes.

On the other hand, agile software development is driven by the notion that the process should move rapidly forward and have minimal need for documentation. In other words, some of the items that are seen as essential in traditional business methods are incorrectly seen as nonessential and perhaps even counterproductive by many in the agile world.

Figure 2-1 shows an example of the gap between traditional business methods and agile practice.

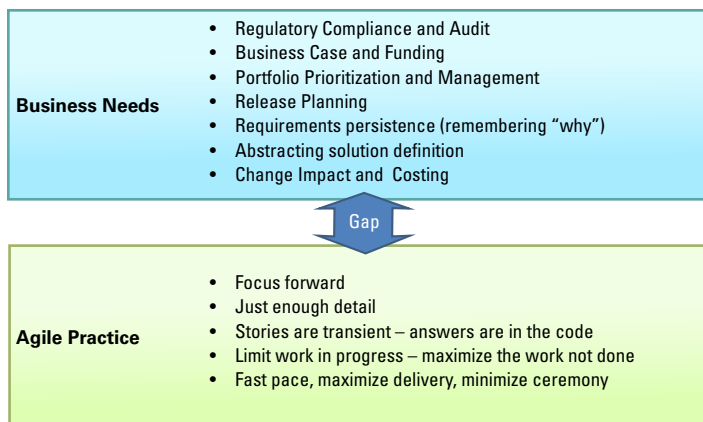


Figure 2-1: Traditional business needs often clash with agile practices.

In Figure 2-1, the traditional business needs shown in the top of the figure typically require considerable advanced planning and documentation. The agile practices shown in the bottom of the figure are focused on getting quick results and minimizing advanced planning and documentation. The gap between the two can be hard to bridge, especially for an enterprise that’s trying to make the transition to agile practices.

Looking at Current Agile Practice

To gain a better understanding of some of the issues companies face when implementing agile practices, you need an understanding of those practices. Figure 2-2 provides a look at the difficulties with current agile practice.

Issues When Attempting to Scale Agile

- User stories are “transient”
- Little organized abstraction
- No auditable tracing
- No analyzing of the business
- No good solution for managing complex needs
- No ongoing record
- Belief that requirements are not needed

Figure 2-2: Some of the difficulties with current agile practice.

The following sections provide you with a better understanding of the points shown in Figure 2-2.

User stories are transient

User stories are what agile practitioners call the information that describes what the application is ultimately supposed to do. For example, a user story might tell the developer that an ATM is supposed to allow bank customers to deposit checks. User stories are typically more generalized than traditional requirements and don’t describe the specific steps needed to reach the goal. User stories do, however, remind us to define the requirement at a later time.

In current agile practice, user stories are meant to facilitate the work of developers—kind of like to-do items—and those stories are discarded after the work is done. In other words, user stories are considered transient rather than persistent documentation. This transient nature can make it difficult for someone to backtrack and figure out the course of development that resulted in the delivered application.

Little organized abstraction

Abstraction is a concept used in the software industry to describe hiding unnecessary complexity based on the target audience for the information. Understandably, different levels of detail are required when communicating with a business user, a developer, or a CIO. A given requirement needs to be described differently for each audience. It is important to provide the proper abstraction for each audience and to organize the abstraction in a meaningful way.

Because agile practice typically includes little or no organized abstraction, this communication to different audiences can be very challenging. This lack of organized abstraction also makes maintaining and upgrading applications in the future considerably more difficult.

No auditable tracing

Because agile practice relies heavily on user stories to determine what needs to be done and the fact that those stories are transient, it's very difficult sometimes to trace back what's being done to higher-level business objectives, needs, and rules. This lack of traceability means that it can be impossible to figure out if the results actually deliver what is required beyond the user view of a very small piece of the system. The agile team may provide a solution that satisfies what they feel was expressed in the user stories, but the solution may or may not completely fit with what's best for the business.

No analyzing of the business

Because agile software developers respond to the needs expressed in user stories, there's often no analysis done of how that story actually correlates with the company's business strategy. This lack of analysis can result in applications that lack necessary features, that may not meet all compliance requirements, or that automate processes that would be more cost effective if they were to remain as manual processes.

No good solution for managing complex needs

Agile practices generally offer no good solution for prioritizing, tracking, or managing releases. This can make handling these functions difficult.

One reason that prioritizing, tracking, or managing releases can be difficult is because, depending on the size of the project, there could be thousands of user stories. Most agile methods imply that there are a few dozen items in the product backlog of future work. As the project proceeds, new items are discovered, whether from the agile teams themselves or from external sources like end users. When the project grows past the point of a few dozen user stories, then grooming the backlog, the process of confirming and prioritizing, becomes very difficult. Further complexity is added if collections of user stories need to be co-released to the end user or must be coordinated with the release of another application.

No ongoing record

Because the user stories that are the basis for development are typically discarded after the work is done, practitioners of agile typically leave no good record of why various things were done except in the code. This lack of recordkeeping can make it difficult to justify elements of whatever final solution is created and lends itself to circular requirements where a system is changed and changed again, and then changed back to where it was previously.

A belief that requirements are not needed

Some people in the agile community feel that formal, recorded requirements definitions are unnecessary. This belief stems from the idea that documentation should be kept to a minimum and that developers should be trusted to produce a good product based on their understanding of user stories. When innovators tried this approach in their enterprise, they often met with success in their trials—or at least close enough

to success for their purposes. However, these trials are often smaller in size, and either don't consider or "gloss over" the business needs identified in Figure 2-1.

Requirements definitions can be extremely important as the projects grow larger, especially in situations where your applications need to comply with certain regulations, which cannot be compromised as they were in the trials. If developers resist the idea of having documented requirements, you may end up with an application that's functional but can't be used because you can't prove that it actually meets all of the business objectives and constraints.

Chapter 3

Understanding the Role of Requirements

In This Chapter

- ▶ Understanding your audience
 - ▶ Making sure you meet compliance requirements
 - ▶ Making good use of components
 - ▶ Taking care of changes
-

Requirements are some of the most important items in making agile practices work for the enterprise. Having a strategy for requirements definition and management (RDM) that meets both the needs of the project and the needs of the business is vital. This chapter looks at some of the reasons why requirements play such a large role in agile for the enterprise.

Understanding the Different Audiences

As you discovered in Chapter 2, user stories are the usual way to communicate what needs to be done to agile teams. In most enterprises, however, the teams are only one of the audiences you need to consider. True, agile teams are an important audience, but they are only one of many, as illustrated in Figure 3-1.

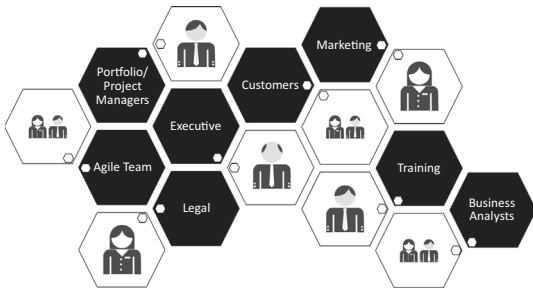


Figure 3-1: Consider all of the audiences.

Typically, you need to accommodate several different audiences:

- ✓ **The agile team:** The developers, customer representatives, testers, and so on who are directly involved in creating the desired solutions.
- ✓ **Portfolio and executive management:** People such as your CEO and other management types who are involved in running the business but not really concerned with details as much as with end results.
- ✓ **The line-of-business:** This is the group that will ultimately make use of whatever solutions are developed. This can vary widely depending on the organization and the type of application, but may include
 - Customers
 - Marketing
 - Training
 - Business Analysis
 - Operations

Different audiences need different abstractions (or descriptions) in order to fully understand exactly what's going on in a project without being burdened by unnecessary detail. For example, your CEO most likely only wants a high level, general overview rather than the complete set of details. On the other hand, your agile teams need to understand in intimate detail any regulatory requirements that must be satisfied in order for your project to pass muster. Another audience might be your legal team who would be concerned with making sure that your applications didn't violate any laws in any of the jurisdictions where you do business.



In many cases, you can simply tweak your project description somewhat for different audiences rather than completely rewrite the description. Reusing as much as possible saves you both time and effort.

Communicating with multiple audiences, such as business stakeholders, executives, managers, and agile teams, is a necessity on software projects. To do this effectively requires that you tailor your message for each audience.

Creating one stable, high-quality perspective of the requirements is difficult enough. To create different perspectives for different audiences is likely to overwhelm the author (typically the business analyst). Requirements automation can really help when you need to create multiple perspectives of the requirements.

Achieving Compliance through Practical Governance

Nearly anything that a large enterprise does today requires meeting specific regulatory standards. Project requirements help define precisely what needs to be recorded and proven for compliance verification, as illustrated in Figure 3-2.

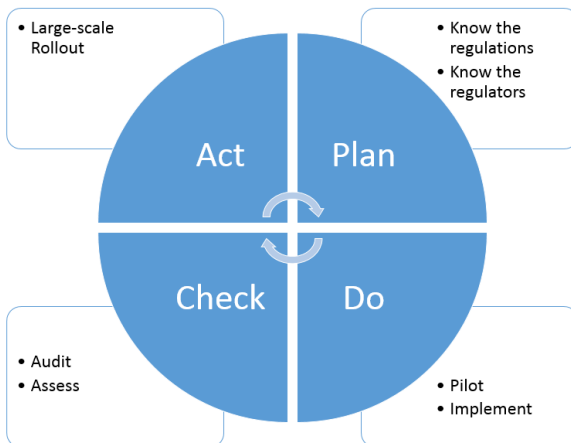


Figure 3-2: Meeting compliance standards through practical governance.

Depending upon your industry, you may need to comply with such regulatory standards as the Payment Card Industry (PCI) Data Security Standard, which details security requirements for members, merchants, and service providers to ensure that any credit card information is processed, stored, or transmitted securely. Obviously, your project requirements would need to specify exactly how you intend to prove compliance with the standard.

Other industries have other compliance needs as well. For example, the U.S. Food and Drug Administration (FDA) has requirements that apply to any food or drug-related items that fall under the jurisdiction of the FDA. Likewise, enterprises that produce aircraft-related products need to meet the Federal Aviation Administration (FAA) flight certification requirements.

No matter what compliance standards your organization must meet, an effective RDM strategy is key to establishing compliance, and later proving that you are in compliance.

Understanding the requirements is also a key factor in applying proper governance to your projects. You use governance—essentially, the policies, roles, responsibilities, and processes established by your organization—to guide how your enterprise uses technologies to accomplish business goals.

In order to apply proper governance, you need to be able to document and prove that your internal standards for software development are being consistently applied. Every enterprise has unique needs and goals that affect how governance should be applied, but in every case, you need to begin with your RDM strategy.

In the case of enterprises, where projects are larger and more complex, where enterprise architecture becomes more prominent, and where oftentimes multiple development teams are involved (not to mention outsourcers, contractors, and adjacent business functions), effective governance is essential for the project to be successful. Because requirements are essentially the blueprints for the project that everyone's work directly or indirectly stems from, the requirements play a central role in project governance.

Supporting Reuse

Constantly reinventing the wheel isn't a very good plan for enterprise efficiency. It simply makes more sense to reuse things wherever possible. It's certainly reasonable to reuse abstract requirements descriptions as well as the components they define for as many purposes as possible, as illustrated in Figure 3-3.

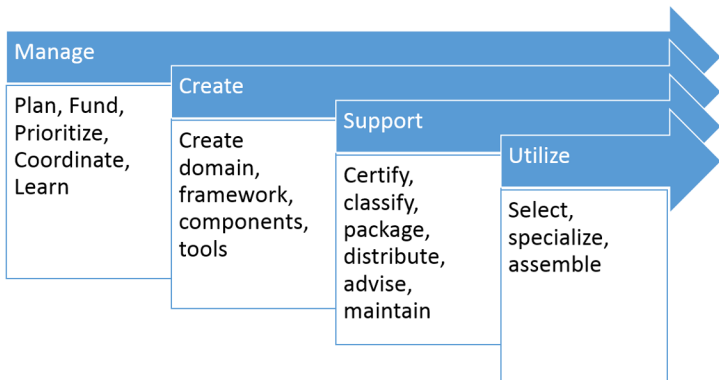


Figure 3-3: Reuse abstract requirements when possible.

Often, the same artifacts can be reused in several different ways. You might, for example, have complete components consisting of requirements, their implementations, and other associated artifacts that can be reused as a package in a Service Oriented Architecture (SOA) approach.

At the very least, common requirements such as business rules, security, and regulatory requirements should be reused as they apply to many or all applications in an enterprise portfolio.

Well-defined and managed requirements are essential to ensure that your applications are developed in ways that are flexible enough so that they can be reused. Requirements definitions need to be managed so that agile teams understand exactly what's needed so that the software they create will actually satisfy all of those needs.

Reuse for savings and efficiency

When requirements are reused, what gets reused isn't just the requirement itself, but all the work that went into its making. This includes the elicitation sessions, reconciling conflicting information, the questioning and feedback, the associated analysis, the creating of trace relationships, the review cycles and approvals, and any subsequent changes or adjustments. All of that work is what goes into the making of a requirement, so to evolve to a point where you have high degrees of requirements reuse can represent substantial savings and drive high levels of efficiency into the process. Reusing requirements, of course, also presents the possibility of reusing the solution-products that derive from that reuse, such as designs, code, components, test assets, and so on. To also be able to reuse these items simply adds to the savings.

Reuse for standardization and consistency

Savings and efficiencies aren't the only reasons for reusing requirements. Reuse can also be mandated from horizontal disciplines, such as architecture, in order to drive consistency and standards across a portfolio of business applications. Consistency and standards across a set of projects make higher-level functions easier to accomplish. Functions such as security, for example, are dramatically easier to accomplish if all applications adhere to a standard approach to authentication and authorization.

Reducing effort and risk

The two biggest benefits of reusing are that you reduce both the effort and the risk involved in your project. The effort is reduced because you won't be re-creating the same things over and over. The risk is reduced because after a component is properly defined, reusing that component means that human error won't introduce unintended changes.

A system such as Blueprint (see Chapter 5) provides the features necessary to support enterprise-class requirements reuse across a portfolio of projects.

Managing Change

A good RDM strategy is also vital when you need to perform an impact analysis to determine exactly how changes in your system will affect users. You need to be able to perform a coarse granularity analysis to determine the impact of a change without interrupting product owners and their teams.

Change, as illustrated by the graphic in Figure 3-4, is inevitable because it's simply not possible to know everything that will happen in the future—and resilience about change is a tenet of an agile approach. If, for example, you had an application that enables users to do online banking through their PC, that application might not be completely suitable for the small screen of a smartphone. Rather than annoying your banking customers by forcing them into an unsuitable compromise, you'd probably want to update your application so that it would support a number of different screen sizes.



Figure 3-4: Resilience is a key tenet of the agile approach.

Seeing an abstract view of change impact

In order to manage changes, you need to be able to see an abstract view of the impact of those changes. For example, if you are developing an online flight-booking application,

you need to consider how something like requiring customers to specify in advance how much checked baggage they'll be bringing would affect the usability of the site. After all, you'd probably find that a certain percentage of your passengers simply wouldn't have that information in advance, so requiring them to provide the information might result in some people choosing to fly a different airline. Although certain departments within your organization might find having that information in advance quite useful, more customer-oriented departments might raise a reasonable objection if they were given the opportunity to review the changes to your website in advance of the changes being implemented.

Estimating the effort of making a change

Changes typically aren't free. Someone needs to put in the time and effort to actually implement a change. You need a system that can help you estimate the amount of effort that will be required in order to estimate the cost of making the change.



It's important to remember that the principles of agile methods tell you that you probably can't know exactly what will be involved when a change is suggested. The best you can hope to do is to make a best effort estimate of how a change will affect your application.

Making better business decisions

Managing change properly also helps you make better business decisions. If you consider the example of requiring passengers to specify the number of checked bags in advance, you can see that doing so might not be a good business decision if doing so results in driving customers to your competition.

To make better business decisions, you need a system that enables you to track and manage changes. The system should include formal methods to manage reviews, comments, and approvals.

Chapter 4

Gaining Advantage with Enterprise Agile Requirements

.....

In This Chapter

- ▶ Understanding roles
 - ▶ Getting to know requirements
 - ▶ Looking at the difference between user stories and requirements
 - ▶ Getting going with your project
-

In Chapter 3, we cover details about requirements. In this chapter, you see how you can leverage requirements to make enterprise agile work in your organization. By combining requirements with agile methodologies, you can eliminate a multitude of issues associated with scaling agile for the enterprise.

Defining Roles

In an enterprise agile environment, people have defined roles as they participate in the development process. Two of the most important roles related to enterprise agile requirements are that of the *product owner* and the *business analyst*. Take a look at how these two roles are defined.

Product owner

The product owner role was originally defined by Scrum (refer to Chapter 1), but it has become mainstream in its adoption

by many agile methods. This product owner role has a responsibility to make the final decisions on requirements definitions, their acceptance criteria, and their priority. Specifically, the product owner is part of the agile team and the business and is the liaison responsible for defining what to do and in what order to do it in. In enterprise agile, the product owner also has an increased emphasis on defining requirements.

The product owner tends to work mostly in an agile ALM (Application Lifecycle Management) system to capture work to be done in user stories. In enterprise agile, the product owner also collaborates with the business analyst role to capture persistent requirements.

Business analyst

The business analyst role is primarily responsible for analyzing the enterprise and for understanding, modeling, and enabling change of systems that support the enterprise. This business analyst role owns the big picture and is responsible for specifying persistent requirements to give a model of the system that lives on after the system is delivered. This role may be outside the agile team but closely collaborates with the product owner role.

The business analyst tends to work to model system requirements for the current and future releases. Collaborating with the product owner, the business analyst also incorporates discovered requirements and finer requirements detail into the agile requirements model as it is discovered during execution of the project iterations. Of special note is that the business analyst role manages the persistent requirements information that lives beyond the end of the project, while the product owner manages the corresponding information in its transient form as a to-do list of work to be done. Over the course of the project iterations, the understanding and detail associated with requirements increase while the items remaining on the backlog decreases.

Figure 4-1 illustrates the roles of the product owner and the business analyst.

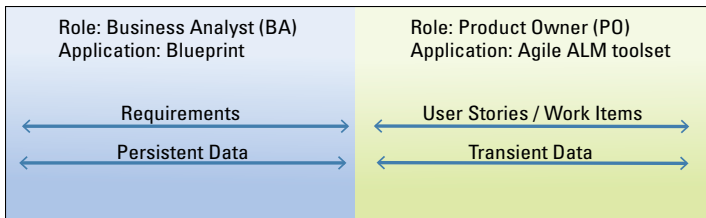


Figure 4-1: The product owner and business analyst roles.

Understanding Requirements in Enterprise Agile

Any project includes a number of different types of requirements. You need to understand how each of these requirements fits into the project planning in order to make best use of them.

Here are several of the types of requirements that you might encounter on an agile project:

- ✓ **Theme:** A theme is a high-level abstraction that often will span multiple applications, or many capabilities of a single application. Themes are usually associated with a release but may in some cases span releases as well. For instance, in the insurance industry you may have a theme to insure new vehicle types where you would need to co-release the applications for writing, billing, and claiming policies even though they are separate applications.
- ✓ **Epic:** Epics are high-level capabilities that are required for the system in order to meet a business opportunity. Because epics are large, it is not expected that they will be implemented in one chunk but that they will eventually be broken into a set of user stories at a later time. Epics are often associated with one application release but may span applications and releases once split into user stories.
- ✓ **Feature:** A feature is used to describe one or more stakeholder needs. Features may be used to bridge the gap between epics and user stories that are sized to fit in a single iteration as they are described in the Scaled Agile Framework, or SAFe. In other agile methods, a feature may be simply known as a large user story.

- ✓ **Illustrative requirements artifacts:** There are a large number of artifact types you can use to specify requirements in a more visual manner. These can include use cases, use case diagrams, user interface mockups, storyboards, flowcharts and activity diagrams, information models, and many more. These artifacts, either alone or in combination, can be used at any level to either provide context for requirements below or provide elaboration for complex areas.
- ✓ **User stories:** User stories are not really requirements, but they are the placeholders that remind agile teams to define the requirements and eventually are pointers to illustrative requirements. User stories may be elaborated by use cases if they are complex enough. User stories are either implemented in one iteration or split into pieces that can each be implemented in a single iteration.
- ✓ **Nonfunctional requirements:** Constraints placed on other requirements are known as nonfunctional requirements. They may be categorized as usability, reliability, performance, supportability, security, regulatory compliance, or business rule requirements to name a few. Each industry will tend to have special kinds of nonfunctional requirements that are used to constrain other, more functional, requirements.

Figure 4-2 illustrates the different types of requirements you might encounter on enterprise agile projects.

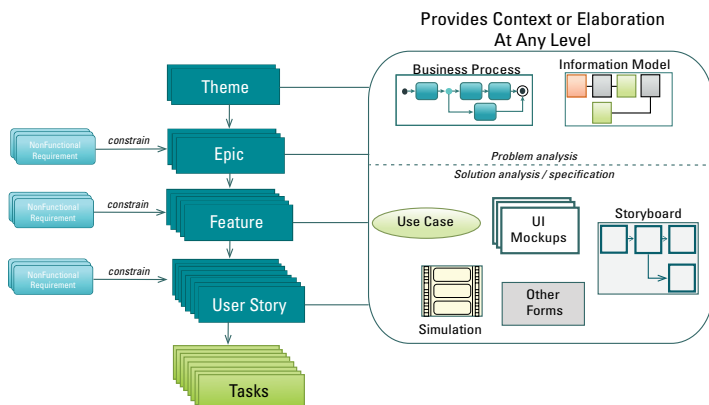


Figure 4-2: Different types of requirements in enterprise agile.

Understanding Why User Stories Aren't Your Requirements

In agile, user stories play a very important role in determining the project's requirements. You might, in fact, believe that user stories actually are the requirements, but this belief is simply not true.

Rather, user stories enable the product owner and the business analyst to collaborate and determine the actual requirements. User stories perform the following functions in an agile project:

- ✓ **Placeholders:** User stories serve as placeholders as the project requirements are being defined. That is, the user stories might say something like online banking customers should be able to check their account balances. In contrast, the actual project requirements might specify a number of detailed steps that spell out exactly how user access must be validated and which databases need to be queried in order to fulfill the customer request. The definition of the actual requirements occurs during execution of the project's iterations.
- ✓ **Reminders for future work:** User stories also serve to remind the business analyst of points within the project where requirements still need to be determined. In this way, the agile practice of prioritization of the product backlog containing these user stories assures that requirements are only defined in detail for user stories that are actually being implemented. It may be that some lower priority user stories will not be implemented for some time—if at all.
- ✓ **Discarded when done:** User stories aren't persistent, but rather they are discarded when the sprint or iteration is completed. The user stories are discarded because they are simply a tool that is used to develop the actual requirements, and the user stories, therefore, serve no purpose once the project is done.

Executing Your Project

Now that you understand the important roles of the product owner and the business analyst, and know a bit more about requirements and user stories, you can begin executing your project. In the following sections, you find out some of the important steps you follow along the way as you execute your project.

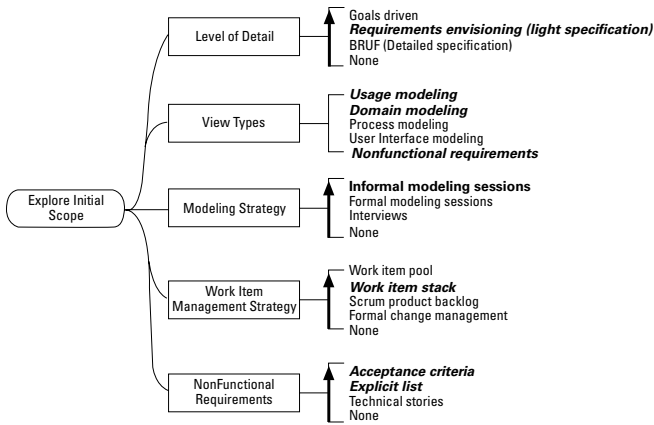
Defining business requirements to win the business case

Most projects begin with determining whether the project makes sense from a business standpoint. Making this determination requires that you first define the business requirements. That is, you need to fully understand exactly what your business needs to do in order to succeed without worrying about the specifics of how the system may be implemented.

Your business requirements might be something along the line of needing to provide customer access to our catalog through our website. The business requirements that you define here don't specify how those requirements will be met, but rather they define the business need.

Your initial pass at defining the business requirements probably won't be the final word on the subject. Rather, it's more likely that this first pass will serve as a conversation starter that enables different clients to contribute to the overall discussion.

Figure 4-3 illustrates the decisions you need to make on how this requirement definition best supports describing system scope for your enterprise using Disciplined Agile Delivery, or DAD.



©2012 Scott Ambler + Associates

Figure 4-3: Exploring the initial project scope and requirements.

Define system requirements during sprint execution

After the business case has been constructed and the decision to proceed with the project has been made, agile methods enable you to define the system requirements as the application is being developed. Basically, the idea is that no one can know in advance exactly what will be needed at every step along the way. Any attempt to predict the distant future in detail will generally result in a great deal of scrap and rework of the requirements as well as adding significant risk.

Agile application development typically proceeds in *iterations* (often called *sprints*), or relatively short bursts of development of small pieces of the overall application. During the sprints, agile team members work with the product owner to learn what is working and what isn't working. Through frequent meetings among team members, an iterative approach to development makes it possible to define the actual system requirements as they are needed and known.

Figure 4-4 provides a look at how the system requirements are defined through this iterative process.

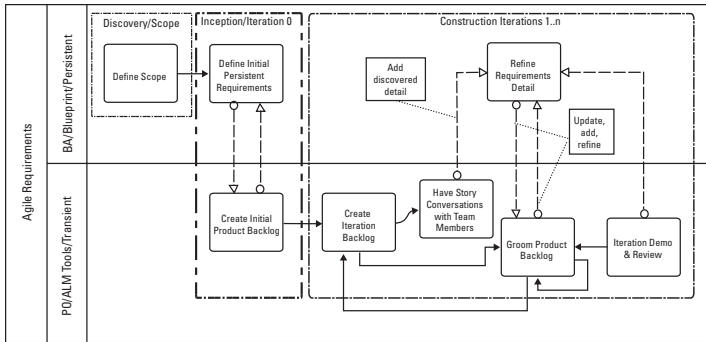


Figure 4-4: Agile requirements are defined as the project proceeds.

Layer-up persistent requirements definition while burning down the product backlog

As the project is being developed, certain requirements become persistent. That is, these requirements are lasting rather than being replaced by new requirements during each sprint.

As persistent requirements are defined, they refine the *agile requirements model*—the requirements that are being completed to finish the project. During the sprint, as each persistent requirement is fulfilled, its corresponding user story is removed from the product backlog. In other words, the product backlog consists of unfulfilled requirements while the agile requirements model consists of fulfilled ones.

Figure 4-5 illustrates how the persistent requirements develop during the project and how the product backlog is eliminated during the process.

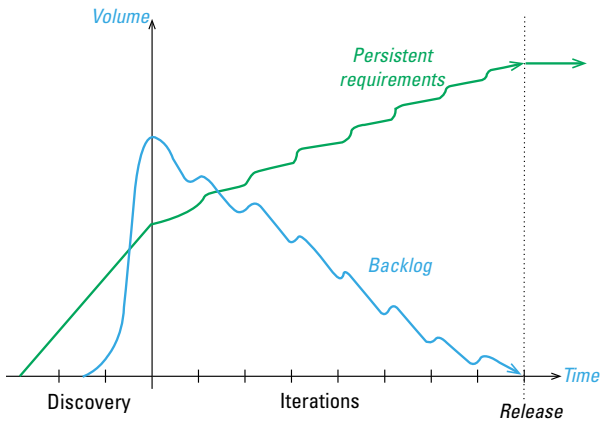


Figure 4-5: Persistent requirements build up as the backlog declines during development.

Just enough detail, just in time

Finally, during your agile project execution, specific details are determined when they're needed. That is, developers aren't loaded down with a whole bunch of information at the beginning of the project, but rather they define the latest information as it's needed.

The idea of just enough detail, just in time, is a key element in understanding agile. Unlike the old *waterfall* development methodology where the entire project had to be planned down to the last detail before any development began, agile assumes that no one can possibly know everything that will need to be done in advance and to attempt to do so results in unnecessary scrap and rework as well as increased project risk.

A large reason why waterfall development evolved to the point where it demanded so much detail early in the process was so that the work of all the business functions outside the immediate project (for example, training, support, operations, legal, marketing, and so on) could proceed, as opposed to being blocked. Of course, much of that detailed information was incorrect—a big part of the problem with waterfall.

However, the need for information early in the process by these adjacent groups remains. Enterprise agile is somewhat of a compromise—providing some information up front to allow others to proceed with their work, and then providing more details iteratively as the project proceeds.

In an enterprise agile environment, the details become progressively clearer as each sprint is completed. This allows the team to be more adaptable to changing needs on the fly, while at the same time allowing business functions to progress forward.

Chapter 5

Using Blueprint for Enterprise Agile

.....

In This Chapter

- ▶ Meeting Blueprint
 - ▶ Using Blueprint for Enterprise Agile
 - ▶ Taking a look at a project
-

In previous chapters, you had a chance to see the important role of requirements and information about enterprise agile. In this chapter, you get to know more about Blueprint, an extremely useful solution for supporting enterprise agile.

Getting to Know Blueprint

Blueprint is one of the most important solution providers for business analysts supporting enterprise agile. Blueprint also happens to be the name of the company's product.

Since its founding in 2004, Blueprint has focused on providing solutions to eliminate the problems associated with defining and managing software requirements. Traditionally, business analysts have used products such as Microsoft Office to manage requirements in the software development process. In fact, some research shows that over 80 percent of business analysts typically use Microsoft Office to define requirements.

Although Microsoft Office works great as a general office productivity suite, it was not designed for defining and managing software requirements. Using Microsoft Office for this purpose makes the process needlessly labor-intensive and error prone

when compared to a specially designed platform, such as Blueprint. Very basic RDM features such as system-assigned, unique identifiers for requirements, requirements properties, or inherent traceability among requirements are not directly provided by Microsoft Office. In addition Microsoft Office does little to support the specific tasks of the business analyst who must elicit, analyze, and specify the requirements, as well as review, approve, track, and report those requirements just to name a few.

Blueprint, on the other hand, combines all of the RDM capabilities needed into a cloud-based application designed specifically for defining and managing requirements using either agile or traditional methods. One way to think of it is that Blueprint combines the functions of Microsoft Word, Microsoft Visio, and Microsoft Excel along with a central, multi-user, versioned repository and social-networking features. These combined features enable you to easily track the status of each requirement as it goes through the development and approval process, and synchronizes requirements information with popular Agile ALM platforms such as Rally, VersionOne, Microsoft Team Foundation Server, and JIRA. Figure 5-1 provides a look at the main capabilities of Blueprint.

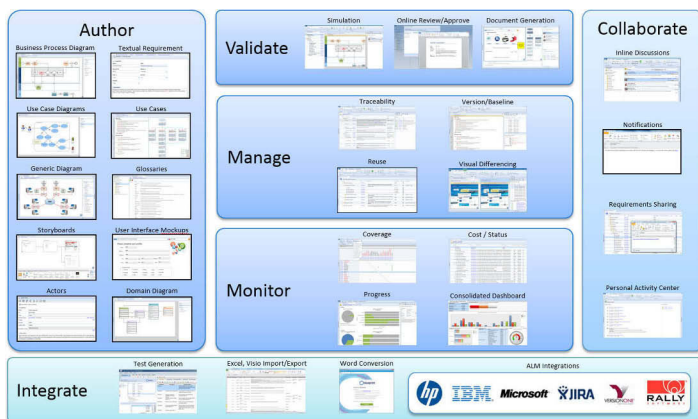


Figure 5-1: Blueprint provides the RDM capabilities needed for enterprise agile.

Blueprint also simulates requirements so that you can check both the appearance and function of the application as it is being developed. In addition, Blueprint generates test cases that can be automatically populated into test management tools to facilitate scenario and user story acceptance criteria testing, among others. These features alone make Blueprint a far better solution than Microsoft Office for enterprise agile requirements.

Applying Blueprint for Enterprise Agile

Blueprint makes it easy for you to embrace enterprise agile. This ease comes because Blueprint is designed to aid a business analyst through the application development process.

With Blueprint, you can easily apply the following to agile practices:

- ✔ **Adapt strengths of traditional processes.** Blueprint enables you to create a hybrid process as needed so that you can use the strengths of traditional processes at the same time you're applying enterprise agile.
- ✔ **Communicate to different audiences.** By incorporating a number of different types of reporting, Blueprint makes it easy to communicate with a broad range of stakeholders by providing them exactly the information they need at the right level of abstraction.
- ✔ **Keep a persistent view of the system available.** Blueprint automatically maintains a record of the current status of all requirements within your project so that you can always see exactly where you stand at any time.
- ✔ **Organize your requirements.** Within Blueprint, your requirements are organized so that you can easily see what needs to be done, what is completed, and the progress through the approval process for each of the requirements.
- ✔ **Define the system being built.** As you use Blueprint, your system definitions are easy to create and access. Blueprint provides you many different ways to visualize the system.
- ✔ **Apply strengths of agile.** Blueprint doesn't lock you into traditional methods, but rather enables you to use the strengths of agile.

- ✓ **Maximize the work not done.** One of the basic principles of agile is that no more work than is absolutely necessary should be done. Using Blueprint, it's easy to see not only which work has been done but also which work remains to be completed, so you can eliminate unnecessary work and thereby maximize the work not done.
- ✓ **Prioritize work.** Blueprint also makes it easy for you to prioritize work because Blueprint enables you to see clearly the relationships between different pieces of your application.
- ✓ **Build the right system.** Blueprint makes it easy to run simulations so that everyone can see how your application will function. This visibility means that building the proper system that handles all of your requirements is far more likely.
- ✓ **Maintain a single platform for requirements modeling.** Blueprint allows you to build a single requirements modeling environment for your enterprise consisting of agile projects, projects transitioning to agile, and traditional projects that share common compliance and other requirements.
- ✓ **Seamlessly integrate with popular agile ALM solutions.** Blueprint integrates with products like Rally, VersionOne, JIRA, and Microsoft TFS, to effectively add a powerful RDM platform that's fully integrated with the agile team's development environment.

Looking at a Sample Project

It's often easier to get a feel for something if you can see it in action rather than just hearing it described. Let's take a brief look at Blueprint as it might be used in a sample enterprise agile project.

Figure 5-2 shows a project structure based on that defined in the Blueprint for Enterprise Agile project template. This structure supports definition of the portfolio, program, and project layers of requirements information to be defined.



Note the trace icons shown at the top right of the figure. These icons indicate that those shapes have trace relationships possibly with other artifacts at the business level, or possibly with system requirements. Clicking the icon in Blueprint reveals the exact list of trace relationships, or alternatively the trace explorer displays a matrix providing a visual mapping of the relationships.

Figure 5-5 shows the Blueprint Traceability Explorer displaying relationships between business-level artifacts and the system requirements that fulfill them. Some of these items are red to indicate they have no corresponding traced requirement (which, of course, is a cause for concern). If user stories are defined within Blueprint, the traceability is extended to those as well. If the stories are defined using a toolset outside of Blueprint, the onus would be on that toolset to provide traceability support to the level of user stories, which is one consideration when defining the process.

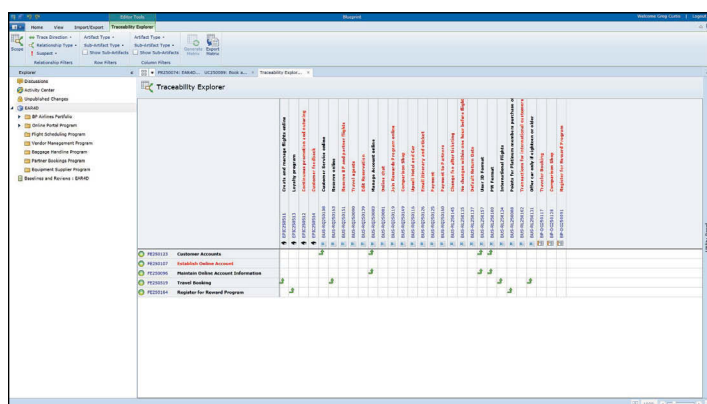


Figure 5-5: The traceability of items in Blueprint.

The screen shown in Figure 5-6 shows Blueprint's online review and approve experience. This is a web-based experience, so all that reviewers need is a browser to access it. Rather than showing all product features, it shows only those specific capabilities needed by reviewers to make it simple to use. From this review and approval experience, users see just those requirements they're being asked to review, provide feedback on, and possibly approve. Discussions, attachments, related requirements, and prior versions of the requirements are all instantly accessible. Reviewers can even run a simulation if they choose.

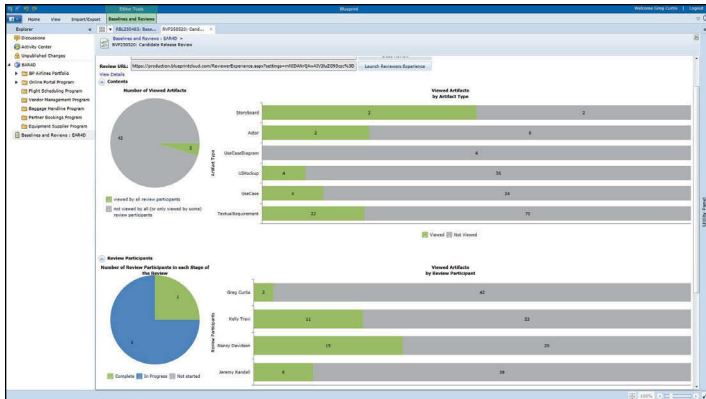


Figure 5-6: Online review and approval in Blueprint.

Blueprint allows large distributed teams to gain control over their requirements and their requirements process. It provides a central repository that holds all requirements-related information and makes it accessible according to easy-to-manage security based on roles. Blueprint was designed for multiple, concurrent users to collaborate on requirements without conflicting or corrupting each other's work. Each requirement's artifact is individually versioned, and prior versions including complete history of who changed what, and when, are available. If users need to know differences between versions, they can select the two versions and instantly compare them to see what changed. Blueprint even shows a side-by-side view, which is indispensable when comparing graphical artifacts (see Figure 5-7).

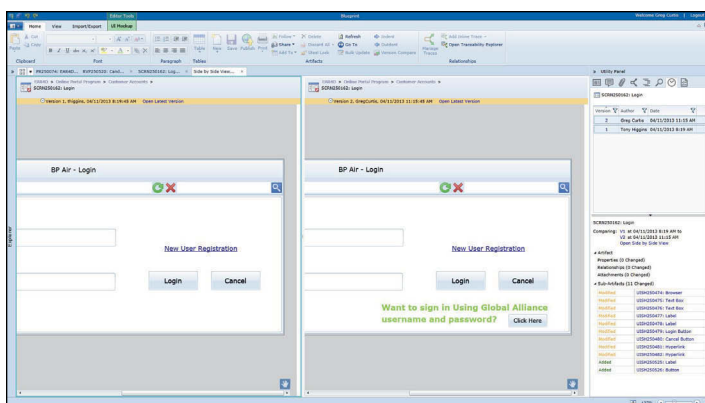


Figure 5-7: Versioning, differencing, and history in Blueprint.



For additional information about Blueprint for Enterprise Agile, see www.blueprintsys.com. You can also find out how to see Blueprint in action and see how it can help your enterprise become more agile.

Chapter 6




Ten Tips for Enterprise Agile Requirements

In This Chapter

- ▶ Using user stories
- ▶ Analyzing your business
- ▶ Doing enough traceability
- ▶ Using a framework
- ▶ Making the best use of technology

This book briefly introduces you to enterprise agile requirements. In this chapter, you go over some of the really important things that you need to know.

Avoid Depending on User Stories Alone

   In Chapter 2, user stories play a big and essential role in helping to define enterprise agile requirements. But user stories aren't the same as requirements. Rather, it's best to think of user stories in the following ways:

- ✓ User stories are placeholders for and pointers to requirements.
- ✓ Requirements can be as simple as acceptance criteria.
- ✓ Requirements can be as complex as a use case model.
- ✓ Use whatever adds value, leave out the rest.
- ✓ Make sure that you always have just enough information.

Analyze the Business



Business analysis is a very big part of enterprise agile. Here are some important things to remember about analyzing your business:

- ✓ Agile isn't a substitute for analyzing the business.
- ✓ You need to clearly understand the business objectives.
- ✓ Business needs are unfulfilled objectives.
- ✓ Business analysis provides the basis for what the business needs.
- ✓ After you've done the business needs analysis, fold in what the users need as part of the business analysis process.

Use Traceability—But Just Enough



Chapter 3 discusses compliance and audit needs and how you need just enough traceability to support those needs. Here are some additional things to consider about traceability:

- ✓ Traceability is needed for a number of reasons, not just to satisfy compliance and audit needs.
- ✓ Traceability is necessary in order to be able to manage the scope of your project.
- ✓ You need traceability to do impact analysis as part of change management.
- ✓ Traceability helps to assess completeness by supporting coverage analysis.
- ✓ People often go overboard. Heed the advice of just enough traceability.

Lean on a Framework



Chapter 4 touches on frameworks for enterprise agile projects. Here are some things you need to remember about using a framework:

- ✓ Frameworks for enterprise agile are proving themselves very useful.
- ✓ Most experts agree that you would be well advised to use a framework.
- ✓ SAFe and DAD are two popular, and largely complementary, enterprise agile frameworks.
- ✓ Blueprint for Enterprise Agile focuses specifically on the requirements aspect of agile at scale, and it is compatible with both SAFe and DAD (as well as project-focused agile methods).

Stagger Your Horizons



In Chapter 3, you find out that an enterprise agile project typically has a number of different audiences, each with its own perspective. As you work through your project, don't forget the following important points:

- ✓ Agile methods force a team to focus on sprints and, to an extent, the upcoming release.
- ✓ You need strategic thinking and consideration for the longer-term business vision.
- ✓ You need to have people with eyes on different horizons:
 - Product owners focus primarily on the iteration, or sprint—what's in it and what's not.
 - Project managers focus primarily on the software release.
 - Business analysts focus not only on the current release but also on the business vision across multiple releases and often across multiple applications.
 - Business architects focus on long-term application sustainability across a portfolio.

Get Visual



Chapter 5 introduces Blueprint and shows you how being visual with your requirements presents a better story than is possible simply using text. Getting visual is an important part

of adopting agile. Here are some important points to remember about how you can best make use of the visual elements:

- ✓ Stories are not enough for enterprise; you also need to define your requirements. A visual representation for these is the best way to do so.
- ✓ You have many, many stakeholders and need to consider communication with all.
- ✓ Visualization has proven itself in every other engineering discipline as the premier way to analyze and communicate complex ideas with multiple audiences and to provide a single, common vision.
- ✓ Remember that simulation is the height of visualization and you need appropriate tooling to support simulation.

Define Your Roles



In Chapter 4, we discuss how different people fulfill different roles in an enterprise agile environment. You want to remember several important things about roles as you adopt enterprise agile:

- ✓ Stereotypical industry roles are morphing and changing.
- ✓ These roles include product owner, product manager, and business analyst.
- ✓ Each role focuses on different things:
 - The product owner's focus is on the development team, prioritizing, helping elaborate stories, and keeping the team moving at velocity.
 - The product manager's focus is on the market, competitive issues, product vision, licensing/pricing, and so on.
 - The business analyst's focus is on the business objectives, compliance, delivering business value from systems, and so on.
- ✓ To avoid disaster you need to clearly define the roles in your organization before the project starts.

Do Just Enough Up Front



As we discuss in Chapter 1, enterprise agile takes a very different approach from traditional methodologies. One of the big ideas of agile is that you should do just enough up front rather than trying to plan everything before you can possibly know all of the facts. Consider these points:

- ✓ Big Requirements Up Front (BRUF) is not the way to go if you want to be agile.
- ✓ Defining no requirements up front is just as bad for different reasons.
- ✓ Doing just enough up front has been proven over the last several years of application.
- ✓ You need to have just the right amount of definition up front to facilitate the necessary business functions, including
 - business case for funding
 - basis for initial project estimation and planning
 - basis for possible outsourcing/contracting
 - basis for risk management and compliance management
- ✓ The *when* changes for enterprise agile requirements as much as the *how*.
- ✓ Most requirements definition occurs in sprint execution.

Make Other Groups First-Class Citizens



As discussed in Chapter 2, you need to consider the needs of your entire enterprise team and treat them as first-class citizens. As you adopt agile, be sure to remember agile has an intense focus on the team. Agile's focus can be to the detriment of other business functions when applied poorly within an enterprise. You need to integrate the extended team

and elevate all business functions to be first-class citizens in the new agile approach. Some of these other functions can include

- ✓ Legal, finance, HR, and other groups possibly affected by the application under development
- ✓ Marketing, sales, support, training, and others involved in exposing and transitioning the application to the user community
- ✓ Operations, support, and others involved in maintenance and sustainment of the application's use while in service

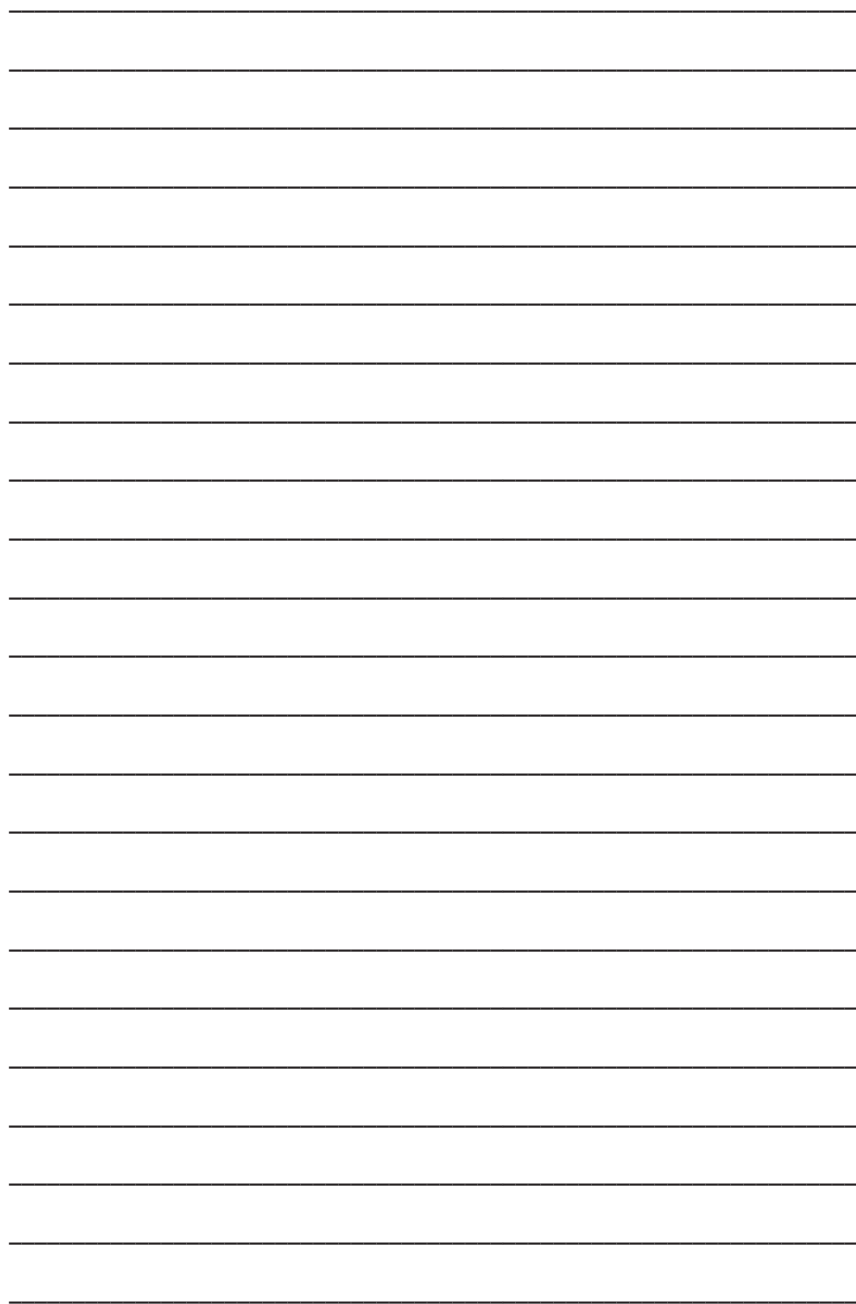
Use Modern Technology



As we mention throughout the book and especially in Chapter 5, you can't really do enterprise agile very well unless you use modern technology to support it. Here are some points to remember:

- ✓ Requirements technology has evolved tremendously in a very short period of time.
- ✓ Many users aren't even aware of this rapid evolution of the technology.
- ✓ Evolution has taken away many of the issues that caused BRUF and waterfall to fail.
- ✓ Blueprint is perfectly suited for use on both traditional waterfall projects as well as projects using enterprise agile.

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.



Blueprint for Enterprise Agile

Blueprint for Enterprise Agile is a solution that provides a solid foundation on which enterprises can employ agile practices on large, distributed, complex business application projects.

Learn more at bit.ly/enterprise-agile

ENTERPRISE AGILE REQUIREMENTS



Position Your Next Software Project for Success

Kick off your software project the right way by getting the best requirement product. The **FREE RFP template** will help you select the right modern Requirements Definition and Management (RDM) application for your business.

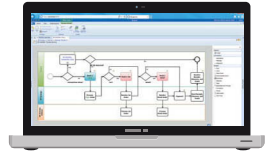
Get it now at bit.ly/rfptemplate

1-866-979-2583 (BLUE) • www.blueprintsys.com



blueprint
The Requirements Company™

SEE BLUEPRINT IN ACTION!



Join Us for Our Monthly Product Demo

See first-hand how Blueprint transforms the Business-IT relationship into a visual and engaging collaboration with less rework and reduced project cost resulting in on-time and on-budget applications.

Check available dates and times at

bit.ly/requirements-demo

1-866-979-2583 (BLUE) • www.blueprintsys.com



blueprint®
The Requirements Company™