GR8PM is a thought-leader in agile project management and systemic enterprise innovation. At GR8PM (pronounced, “Great PM”) we believe that successful outcomes maximize customer value for both the end-user customer and the organizational customer!

- We view Traditional, Agile, and Hybrid project management frameworks as complementary tools that every professional project leader must know. Selecting and using the right framework is a significant first step to delivering successful results.

- GR8PM knows that passing exams does not make a successful project leader…but we also know that exams demonstrate proficiency with core concepts, generally accepted standards, and the industry’s lexicon. In the absence of a better approach many employers and clients look to certifications for assurance that they are being served by well-trained professionals.

- This world-class reference serves students pursuing designations as a PMI Agile Certified Practitioner (PMI-ACP®) or a Scrum Alliance Certified Scrum Professional or both. If you choose to take them, this book guarantees you pass both exams. And even more importantly, assures your success using agile methods as a professional project leader!

Includes In-Depth Coverage and Best Practices For:
- Scrum
- eXtreme Programming (XP)
- Lean Software Development (LSD)
- Feature Driven Development (FDD)
- Agile Unified Process (AUP)

Also Includes Best Practices For:
- Crystal
- Dynamic Systems Development Method (DSDM)
- Test Driven Development (TDD)
- Agile Modeling

Exam Preparation Aids Include:
- Chapter Practice Tests
- Terminology Matching
- Crossword Puzzles
- Word Searches

The PMI-ACP® certification is the fastest growing certification in PMI history! The Certified Scrum Professional is the next step for serious Scrum Masters. This book helps students pass both exams on the first try…100% money-back guaranteed! No other reference offers a better guarantee; so why gamble your time and money?!?

After you pass your exam this book also serves as a comprehensive desk reference for handling everyday agile project management challenges. No other reference has everything in one place for you!

- Everything needed to pass the PMI-ACP® Exam
- Everything needed to pass the CSP Exam
- Practice Tests with 425 Questions and Answers
- Access to FREE TRIAL of GR8PM Online Exam Simulator

Book Includes:

- PMI-ACP® and Certified Scrum Professional Exam Prep and Desk Reference
- Comprehensive Resource for Project Management’s Top Certifications

PMI-ACP® and CSP Desk Reference

John Stenbeck, PMP, PMI-ACP, CSM, CSP

www.gr8pm.com

Photo: Federal Highway Administration, Central Federal Lands Highway Division (FHWA/CFLHD)
PMI-ACP and Certified Scrum Professional Exam Prep and Desk Reference

Comprehensive Resource for Project Management’s Top Certifications

John G. Stenbeck, PMP, PMI-ACP, CSM, CSP

First Edition

GR8 PM
Traditional. Agile. Hybrid.

La Mesa, CA
Acknowledgements

To my sons, JT and Michael, who inspire me, thank you!

To my Cursillo brothers who have cheered me on, please keep praying for me.

We would also like to acknowledge and thank:

- Kensington Type & Graphics’ Tamara Parsons for her creative genius and diligent follow-through.
- Lauren Mix for her tireless editing, over and over again…
- JT and Lindsey for the endless task of keeping GR8PM running so this edition could actually be done!

Your contributions were invaluable in making this project a reality.

100% Money-back Guarantee

At GR8PM we believe you have the right to know that the information resources you trust are the very best. We also believe that the truest and simplest way to demonstrate our expertise in helping you prepare for either the PMI-ACP® or the CSP exam is to offer you a 100% Money-back, 1st Time Pass, Guarantee! No other author, company or resource demonstrates the same confidence in their product. We know this book contains everything you need to pass. We believe you will use it to adequately prepare and we trust you! So if you don’t pass the exam the first time - for any reason - within 180 days of purchasing it, just send us the original notice from Prometric showing you did not pass, and your purchase receipt, and we will send you a refund…no questions asked!

Simply the best guarantee available!
## Contents at a Glance

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Agile Project Management Value Proposition</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Introducing Agile Project Management</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Initiating Projects</td>
<td>61</td>
</tr>
<tr>
<td>4</td>
<td>Planning Projects – Part 1</td>
<td>95</td>
</tr>
<tr>
<td>5</td>
<td>Agile Estimating</td>
<td>125</td>
</tr>
<tr>
<td>6</td>
<td>Planning Projects – Part 2</td>
<td>145</td>
</tr>
<tr>
<td>7</td>
<td>Iterating Projects</td>
<td>175</td>
</tr>
<tr>
<td>8</td>
<td>Controlling Projects</td>
<td>215</td>
</tr>
<tr>
<td>9</td>
<td>Contracting and Accounting Control</td>
<td>235</td>
</tr>
<tr>
<td>10</td>
<td>Closing Projects</td>
<td>257</td>
</tr>
<tr>
<td>11</td>
<td>How to Pass the Exam...The First Try!</td>
<td>279</td>
</tr>
<tr>
<td>12</td>
<td>Comprehensive Exams</td>
<td>305</td>
</tr>
<tr>
<td>GLOSSARY</td>
<td></td>
<td>365</td>
</tr>
<tr>
<td>INDEX</td>
<td></td>
<td>385</td>
</tr>
</tbody>
</table>
# Table of Contents

**CHAPTER 1**

**Agile Project Management Value Proposition** ............................................... 3
Agile Value Proposition… in a Nutshell ............................................................... 3
Ethos of Agile Project Management ..................................................................... 4
Agile Manifesto and the 12 Principles ................................................................. 6
What’s In It For You? ............................................................................................. 7
Should You Pursue the PMI-ACP® and Scrum Alliance CSP? ......................... 8
Why THIS Book? .................................................................................................... 9
What the Exams Test ........................................................................................... 9
About the Authors ............................................................................................... 10
Conventions & Standards Used in this Book ..................................................... 11

**CHAPTER 2**

**Introducing Agile Project Management** ......................................................... 15
Chapter Highlights ............................................................................................. 15
Overview of the Agile Project Management ......................................................... 16
Is Agile Really Needed? ...................................................................................... 18
Mapping Agile to the PMBOK® .......................................................................... 19
Agile Planning and Estimating .......................................................................... 20
Agile Execution and Control ............................................................................. 21
The Origins of Agile Project Management ......................................................... 22
Application of Lean thinking to Project Management ..................................... 24
Summary of Agile Frameworks & Tools .............................................................. 31
1. Scrum .............................................................................................................. 32
2. Extreme Programming (XP) .......................................................................... 33
3. Lean Software Development ......................................................................... 36
4. Other Frameworks ........................................................................................ 37
5. Test Driven Development (TDD) .................................................................. 43
6. Agile Modeling (AM) .................................................................................... 44
The Agile Process Map™ & Agile PM Processes Grid™ ..................................... 45
Chapter Close-Out and Exercises ..................................................................... 47

**CHAPTER 3**

**Initiating Projects** ......................................................................................... 61
Chapter Highlights ............................................................................................. 61
Overview of Agile Project Initiation .................................................................... 62
Identify Stakeholders and Build Engagement .................................................... 63
Identify Stakeholders ......................................................................................... 64
Build Stakeholder Engagement ......................................................................... 65
Value-Driven Delivery ....................................................................................... 68
Document the Business Case ............................................................................ 68
Write Contracts ................................................................................................... 69
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Planning Projects – Part 1</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>Chapter Highlights</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>Understanding Agile Planning</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>External Stakeholders Engagement</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>Designing the Product Roadmap</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>Identifying Minimal Marketable Features (MMF)</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>Prioritizing Stories for the Project Team</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>Value-Driven Delivery</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>Release/Iteration Plans</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Planning Activities</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>Decomposition and Progressive Elaboration</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>Story Maps</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>Adaptive Planning</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>User Stories</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>Iteration Backlog</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>Definition of Done</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>Chapter Close-Out and Exercises</td>
<td>112</td>
</tr>
<tr>
<td>5</td>
<td>Agile Estimating</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>Chapter Highlights</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>Estimation and Sizing</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>Estimation and Sizing Risks</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>Wideband Delphi</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>Planning Poker</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td>Affinity Estimates</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td>Chapter Close-Out and Exercises</td>
<td>133</td>
</tr>
</tbody>
</table>
# Planning Projects – Part 2

- Chapter Highlights ............................................ 145
- Team Performance ............................................. 146
- Coaching/Facilitation ........................................ 146
- Collaboration/Negotiation .................................... 147
- Motivation/Empowerment .................................... 148
- Risk Management .............................................. 148
- Risk-Adjusted Backlog and Regulatory Compliance .... 148
- Communication .................................................. 149
- Communication Protocols .................................... 149
- Information Radiators ........................................ 149
- Team Space ...................................................... 151
- Agile Tooling ...................................................... 152
- Continuous Improvement .................................... 155
- Value Stream Mapping ....................................... 156
- Cross-Functional Team Formation ......................... 157
- Metrics Identification ......................................... 158
- Chapter Close-Out and Exercises ......................... 162

# Iterating Projects ............................................. 175

- Chapter Highlights ............................................ 175
- Overview of Execution ....................................... 175
- Product Backlog Grooming .................................. 177
- Improve Value-Driven Deliverables ......................... 179
  - Measuring and Using Cycle Time ......................... 179
  - Using Cumulative Flow Diagrams (CFDs) ............. 183
- Facilitate Adaptive Planning ................................ 185
  - Create Visual Controls ...................................... 185
  - Apply Test-driven Practices ............................... 190
  - Agile Modeling ............................................... 192
  - Wireframes .................................................... 192
- Nurture Team Performance .................................. 193
- Enhance Risk Management .................................. 194
  - Simplify Problem Solving .................................. 195
  - Incorporate Continuous Integration .................... 195
  - Risk-based Spike ............................................ 196
  - Risk Burn-down Charts .................................... 198
  - Verification and Validation ................................. 198
- Adapt Communications ........................................ 199
  - Daily Stand-up, IterationReview, & Team Retrospective Meetings .......................... 199
  - Osmotic Communication .................................... 201
- Implement Continuous Improvement ....................... 202
## Table of Contents

### CHAPTER 11

- **How to Pass the Exam...the First Time!** .......................... 279
  - Chapter Highlights .................................................. 279
  - Integrate Professional and Social Responsibilities ................. 280
    - Recognize Professional and Social Responsibilities ............. 280
    - The PMI Code of Ethics and Professional Conduct ............... 280
    - The Four Values – Responsibility, Respect, Fairness, and Honesty . 281
    - Endorsing Responsibilities to the Profession ..................... 282
    - Safeguarding Responsibilities to Customers and the Public ...... 282
    - Observing Other Responsibilities ................................. 283
  - How to Pass the Exam...the First Time ......................... 283
  - The Key is the 5 Ps .................................................. 284
  - What The Exam Tests ................................................. 285
    - A Passing Grade .................................................... 285
    - Using a Testing Strategy ......................................... 286
  - Getting to the Test ................................................... 288
    - Fill Out the Application ............................................ 288
    - Application Requirements ....................................... 288
    - The Testing Environment ........................................ 289
  - After Taking the Exam ............................................... 290

### CHAPTER 12

- **Comprehensive Exams** ............................................. 305
  - Exam 1 ............................................................. 306
  - Exam 2 ............................................................. 327
  - Answers Exam 1 .................................................... 348
  - Answers Exam 2 .................................................... 356
  - **Glossary** ......................................................... 365
  - **Index** .......................................................... 381
Agile Value Proposition ... in a Nutshell!

The playing field where human communities compete, from companies to nations, requires innovation and agility to create or maintain any competitive economic advantage. That sustainable advantage comes from systematic innovation. A strong correlation of that fact can be seen for industries from consumer electronics and pharmaceuticals to telecommunications. A mild correlation can also be seen for industries from healthcare and insurance to retail/wholesale.

So whether communities thrive, survive, or fail is directly correlated to their ability to be agile and innovative. That core driver is behind the ever increasing demand for Agile Project Managers. That is also the agile value proposition... in a nutshell!

Stated another way, projects allow organizations to operationalize their innovative strategic vision. Executing those projects using the correct project management framework insures those projects maximize the positive impact of the assets and people deployed to deliver them. To accomplish this, professional project managers are increasingly being tasked by their organizations to synthesize the best practices of traditional and agile frameworks into an approach that is tailored to the environmental demands facing them.

Without a solid base of Agile Project Management knowledge it is impossible for a project manager to fulfill that responsibility effectively.
Evidence strongly suggests that the future of project management is running **hybrid projects**. Tomorrow’s professional project manager cannot be effective without the ability to run hybrid projects. So, in a nutshell, the Agile Project Management value proposition is the ability to manage hybrid projects because you have an understanding of both traditional and agile frameworks!

![Figure 1.1 Distribution of Projects being Managed using Various Frameworks.](image)

**Ethos of Agile Project Management**

When we consider the quantitative, engineering environment where most project management takes place, it might seem odd to use a word like ethos. Ethos conjures up other words like philosophy, culture, and attitude, but there is a good, solid quantitative reason for understanding, accepting, and applying the ethos of Agile Project Management.

That reason is **risk management**. A project cannot survive, much less thrive, without effective risk management, which for any project, hinges on cross-functional teams communicating effectively, a critical variable.

The agile ethos recognizes this fact as being equivalent to gravity. You can’t escape it so why deny it. Instead, leverage it to create potential advantage. **The agile ethos is founded on the principle that teams must be cross-functional, can be trusted, and should be collocated, in order to optimize the chance to identify risks and reduce errors.** Such a team will experience osmotic communication about risks that are obvious to one team member, but perhaps not to other team members – often the ones that most dramatically affect the success of the project.

Unfortunately, good communication does not automatically occur in any environment and must, instead, be cultivated as part of the ethos and framework of Agile Project Management.

This basic fact has been observed hundreds of times during a simple exercise, called the “Stare and Share,” conducted at numerous training seminars. The exercise has three steps. In step 1, all participants are asked to stand, choose a partner, and face the partner. They are then given 20 seconds (which always seems like a lifetime to the participants) to “visually study their partner.” In step 2, they are instructed to stand back-to-back (so they cannot see each other) and to change three things about their appearance. In step 3, they are asked to turn so they can each see their partner once again, and then identify the three changes their partner made. At no time are the
participants told not to collaborate, and they are even referred to as “partners” repeatedly throughout the instructions. Nevertheless, there are normally less than one-third of the participants who identify all three changes, and less than five percent who ask for, or reveal, the changes to one another.

This human trait impacts many project teams. Critical information - information vital to success - will not be shared automatically, unless the people assigned to the project unite as a team. Commitment to one another’s success does not begin until a team is born! Catalyzing as a team, therefore, is critical to success and only happens when communication is properly facilitated.

So the quantitative goal of having collocated, a cross-functional, trusted team is to reduce project risk by reducing the unknown about the project. The situation before a team catalyzes can be seen in Figure 1.2. The illustration shows that as long as the project is worked on by a group of people who have not become a “team,” the relationship between what is known and unknown remains static. That produces a significant “blind spot,” where things that are not known by the PM overlap with things that are not known by the team (i.e., the bottom right quadrant).

The situation after a team catalyzes can be seen in the next illustration, Figure 1.3. When a team environment exists, everyone is committed to being responsible for the project’s success or failure. That is because relationships formed when the team made the hard commitment to the iteration goal (a process that will be explained in Chapter 4). Those relationships engage everyone to remain vigilant, watching for areas of risk “owned” by other team members - as naturally as soldiers covering each other’s back in combat. Most importantly, the automatic disclosure of observations, insights, and information occurs.

![Figure 1.2 Risk Before Team Formation.](image1)

![Figure 1.3 Risk After Team Formation.](image2)

This reduces the resulting blind spot, increasing the general level of knowledge about the project. The way this happens can almost be described as triangulation. As members of the team share their knowledge, other members can integrate that with their knowledge base and then share their own observations and insights. In the process, risks that had been unrecognized, and therefore unknown, are identified by triangulating their position. This occurs much like the navigation of early mariners using a sextant and the stars. Two data points are used to help identify a third. Precisely because trusted, cross-functional teams supply a mix of skills, risk and errors are reduced. So risk is diminished and the opportunity for success is enhanced. In the agile ethos, empowering teams is central to realizing the goal of a successful project.
Another facet of the agile ethos is commitment to a culture of **continuous improvement**. The commitment to continuous improvement shows up in many of the practices, ceremonies, and tools used in Agile Project Management in order to ensure long-term sustainable benefits which translate into competitive advantage for the customer, not just quicker project deliverables in the short term.

The agile ethos was expressed originally in the Agile Manifesto and then expanded in the Principles behind the Agile Manifesto (i.e., “The 12 Principles”).

**Agile Manifesto and the 12 Principles**

One of the seminal events in the rise of Agile Project Management occurred in February 2001, at the Snowbird resort in Utah. Seventeen luminaries in the field of software development met to discuss the need for alternatives to the failure-prone project management processes then being used. Perhaps no one was more surprised than the participants when they achieved a meeting of the minds and all agreed to sign the Manifesto for Agile Software Development, now commonly referred to as the Agile Manifesto.

The group named itself *The Agile Alliance* and published the Manifesto for Agile Software Development *(See below)*. It outlines the fundamental beliefs that reinforce agile software development, a precursor to Agile Project Management.

**Manifesto for Agile Software Development**

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.

Notice that the central word in the Agile Manifesto is “over.” The manifesto does not support the common, erroneous interpretations that it supports individuals and interactions “instead of” or “rejecting” processes and tools. Great damage has been done to the value of Agile Project Management by imposters claiming their focus is working software “not” comprehensive documentation, or customer collaboration “without needing” contract negotiation, or responding to change “without” following a plan. Proponents of such approaches are simply not agile, in spirit or in fact!

The Agile Alliance also published the *Principles behind the Agile Manifesto*, which stated the following.

*We follow these 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, then tunes and adjusts its behavior accordingly.

While the ethos of agile, as imbued in the promise of the Manifesto, is laudable and includes some serious wisdom, the Manifesto, the Agile Alliance, and the many organizations it spawned have not fulfilled the potential of agile. In a certain sense they became a victim of their own success.

Basically, the insights that human beings are far more than biological machines and that the solution to our complex problems won’t be found in the level of thinking that got us here (to paraphrase a famous quote attributed to Einstein), are absolutely valid and form the core of the agile ethos.

The resulting problem becomes that agile has been misinterpreted and misused to justify all kinds of bad thinking and bad behavior under the banner of letting teams “be productive.” Some agile evangelists have suffered from a myopic viewpoint that lost sight of the need to be able to do things at the enterprise, portfolio, and program levels. It seems that their fear of slipping back down the path of the old command-and-control approaches has led them to believe it is a dichotomous choice. If the choice were between using agile teams or submitting to command-and-control processes their position might be reasonable. However, the best solution to the situation is not “either/or,” the best solution is “both/and.”

Holding onto the ethos of agile and using the insight that Lean provides, allows for a framework and processes that are an intelligent balance for flexible, effective teams and long term organizational stability. This understanding comes into focus when the beliefs, principles, and paradigms on which a framework is built, are carefully examined and appropriately applied.

As you study for the ACP exam, we hope your goal will be to embrace the ethos of agile and to employ it with the maturity that your studies should stimulate.

What’s In It for You?

Acquiring the industry’s leading certifications provides the opportunity to lead projects that have the WOW factor. It can give you the chance to lead dream teams on projects of a lifetime. Additionally, if you feel like your job or your future is at risk, being certified can keep you from getting passed over instead of promoted, at the next opportunity.

Additionally this book helps you unlock your career potential because it is more than just an exam preparation book. It is a Desk Reference for Agile Project Management Frameworks. It serves as a handy source of “how to” knowledge and best practices as you lead your teams and organization to the next level of agility and competitive advantage!
Should You Pursue the PMI-ACP® and Scrum Alliance CSP?

Yes …and YES! Every project leader will benefit from knowing how and when to use Agile Project Management tools. This book is for:

- Program Managers and Product Owners
- Project Managers and Scrum Masters
- Project Team Members – Engineers, Analysts, Developers, and Testers
- Customers, General and Senior Management
- College and University Instructors

<table>
<thead>
<tr>
<th>Frameworks Covered:</th>
<th>Agile Certified Professional (PMI-ACP®)</th>
<th>Certified Scrum Professional (CSP)</th>
<th>Certified Scrum Master (CSM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scrum</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XP</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lean</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other Frameworks</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Agile PM Education
- 21 hours; Taught by any PM education provider
- Be current holder of CSM, CSPO, or CSD certification
- 16 hours; Must be taught by Certified Scrum Trainers

Work Experience
- 2,000 PM hours; 1,500 agile hours (Non-overlapping)
- 2,000 hours of Scrum-related work in the past two years
- No requirement; 2 day class

Industry Orientation
- Industry neutral
- Software development
- Software development

Exam Details
- 120 questions 3 hours Pass/Fail
- 150 questions 3 hours Pass/Fail
- 35 question online evaluation, 5 attempts to pass allowed

Exam Administration
- Proctored by a PMI contractor
- Proctored by a Scrum Alliance contractor
- Class instructor

Credential Expiration
- 3 year cycle; Must earn 30 PDUs in agile PM per cycle
- 2 year cycle; Continuing education required to retain certification
- 2 year cycle; Must retake evaluation to retain certification

Figure 1.4 Comparison of the PMI-ACP® and Scrum Certifications.

Why THIS Book?

At GR8PM (pronounced “Great PM”) we view Traditional, Agile, and Hybrid project management frameworks as complementary tools that every professional project manager needs to know. Selecting and using the right framework is a significant first step to delivering a successful project. We also know all successful PMs are pragmatists and once they have good command of the framework and its tools, they will make the right choice in applying the best approach to leading their project to success!

GR8PM also knows that employers and client organizations want the assurance of well trained professionals in their organizations. Therefore, we created this world-class exam preparation book for the new Agile Certified Practitioner (PMI-ACP®) and went a step beyond. In addition to doing an in-depth analysis of all 11 books listed by PMI as the basis of the test, we used our own expertise to create a useful, accessible digest of the Project Management Agile Body of Knowledge™
That means this book will help you pass the exam and even more importantly, succeed in using agile methods as a professional project manager!

With our experience and analysis reduced to writing, we applied the concepts of Accelerated Adult Learning Theory to the instructional design of this book to ensure it would give students the most help possible.

In a nutshell, Accelerated Adult Learning Theory, which was pioneered around 1980, confirmed what most people would see as common sense. Specifically that **people learn in three different ways – visual, auditory, and tactile.** That research also confirmed that engaging all three ways creates the greatest learning and long-term memory retention. Each way of learning, or learning modality, is like a different type of radio broadcast – AM, FM, or Satellite. You find different music and different programs in each type of broadcast and tune to those stations for different reasons. Similarly, people use different ways of learning in different situations. Sometimes, visual learning works best. Other times, auditory learning is the best approach. And in some situations, tactile learning is needed. Often, how energetic or how tired an individual feels will impact which type of learning works best. Without a doubt, the absolutely best and longest-lasting type of learning takes place when all three ways are stimulated.

This book is designed to capitalize on the findings of Accelerated Adult Learning Theory and engage you in visual, auditory, and tactile ways of learning. Specifically, you will find explanations, illustrations, suggestions for writing notes and flashcards, and test questions throughout this manual. The explanations and activities provided will stimulate each type of learning modality and result in a thorough, long-lasting understanding of the material and concepts of Agile Project Management.

**About the Author**

**John G. Stenbeck, PMP, PMI-ACP, CSM, CSP,** is the President of GR8PM, Inc. His particular value comes from a combined background in Accounting, Operations, I.T. and Project Management.

As the Project Cost Management Office manager at an aerospace company John developed deep expertise in applying the Federal Acquisition Regulations (FARs) and the Defense extension of the Federal Acquisition Regulations (DFARs) to large multi-year and multi-generation programs and projects. Because he had primary responsibility for handling audits by the Defense Contract Accounting Agency (DCAA) and the Defense Contract Management Auditors Office (DCMAO) his expertise has been battle tested.

He has demonstrated the ability to manage large, complex projects to success where others have failed. John has extensive experience implementing enterprise resource planning (E.R.P.) systems at firms in the aerospace, shipbuilding, and construction industries. The companies share a business process environment that can be categorized as engineer-to-order (ETO) or configure-to-order (CTO).

John is also a respected author, speaker and trainer. He has authored seven nationally and internationally delivered training courses for project management practitioners in addition to this highly acclaimed book.

As a speaker at the 2008 PMI – Global Congress, John set the all-time attendance record for an Area of Focus presentations when the first audience of 640 made it a **sell-out!**
Keys of Estimating that Successful Project Managers Can’t Live Without! continues to be popular at PMI Chapter events and meetings. Interestingly, at the 2010 Global Congress also John set the second highest-ever record – over 400 – with Agile Project Management Mastery in 60 Minutes…Guaranteed! It has become equally popular at PMI Chapter events and meetings.

John is an Adjunct Instructor for the University of California – San Diego’s School of Extended Studies, where he teaches project management in the Systems Engineering Certificate program. As a trainer, he has taught numerous public and corporate on-site programs to over 8,000 students. John helps technical professionals master project management and executive leadership skills.

John has done extensive consulting and coaching in traditional project management, agile project management, extending agile to the enterprise, and leadership, team building, and communications.

A partial list of John’s clients includes:

- Booz Allen Hamilton, Inc., McLean, VA
- Guinness Bass Import Company, Greenwich, CT
- Lucent Technologies – Bell Labs, Allentown, PA
- Nike Corp., Beaverton, OR
- Oracle Corp., Redwood Shores, CA
- Orange County Public Works, Orange, CA
- Qualcomm Inc., San Diego, CA
- U.S. Army – Space & Terrestrial Comms., Fort Monmouth, NJ
- U.S.D.A. – National Finance Center, New Orleans, LA
- Visa – Smart Cards, Foster City, CA

John is certified by the Project Management Institute (PMI) as a Project Management Professional (PMP®) and an Agile Certified Practitioner (PMI-ACP®) and by the Scrum Alliance as a Certified Scrum Master (CSM) and a Certified Scrum Professional (CSP). Moreover, he has an ITIL v.3 Foundations certification.

John is a Past President of the PMI – San Diego Chapter where he served as the Vice-President of Professional Development before becoming the President. He is involved in a number of professional user groups outside of the project management field. He graduated from San Diego State University with a BS in Business Administration, with an emphasis in Accounting.

Beyond professional and career pursuits John is a passionate supporter of youth soccer where he has been a coach and referee for many years. He was a California licensed Soccer Coach and certified by two different officiating bodies as a referee.

Personally, John has been active in men’s ministry as part of the Cursillo Community of San Diego, prison ministry as part of Kairos Council of San Diego, supported the youth ministry efforts of San Diego’s All for God Steubenville Conference for many years.

His hobbies include cooking, swimming, and enjoying the outdoor San Diego lifestyle with his two sons and many friends.

John is available to speak to your company or group on a variety of topics, including:

- Why Agile is Mainstream!
- Rolling Out Enterprise Agile Practices (Advanced Topic)
• Five Keys to Estimating Every PM Needs!
• Three Vital Steps to Mastering Risk Management
• Transform Your Career Network from Unreliable to Championship!!
• Unlocking Your Project Leadership Potential…NOW!
• Agile PM Mastery in 60 minutes… Guaranteed!
• Conquering Project Cost Management with a Simple, Proven System
• Successful Project Scheduling… In a Nutshell

John’s programs return you to work…fine-tuned and ready to take action. He works with you...and for you! At the core, he delivers practical, measurable solutions to meet your real needs...both now and in the future!!

Conventions & Standards Used in this Book

Chapter Highlights
At the beginning of each chapter, you will find an overview of the significant information introduced and covered within. You can use the summary to identify the critical content you need to know before you proceed or as a quick reference guide in the future.

Chapter Endnotes
The extensive endnotes included in this ACP Exam Prep book allow it to also be used as a Desk Reference for the Project Management Agile Body of Knowledge™, helping you locate useful source information. Although the endnotes may only be marginally helpful for exam preparation, they will be invaluable long after the pain of the exam has faded! That is when your boss and coworkers will expect you to be an expert in all things agile, and you will refer to the source material contained in the endnotes time and time again.

Key Notes
Throughout the book, the Key Note icon (shown on the left) is used to call your attention to important facts to master for the exam, as well as for daily practical reasons.

Flashcard Moments
Also throughout the book, we will use the Flashcard icon (shown on the left) to call your attention to important facts that are a good choice for creating flashcards. As we mentioned before, Adult Learning Theory has shown that learning through various channels increases retention and mastery. The act of writing a flashcard impacts learning on two channels – writing and then subsequent review. You can also buy pre-printed flashcards on-line at our website.

Agile PM Processes Grid™ Exercise
One of the best test preparation exercises a student can do is the Agile Project Management Processes Grid™ exercise. It gives students a chance to see how much content they have retained and can process from memory onto a blank sheet of paper, which will be provided by the proctor at the testing site. At the end of each chapter, you should take a blank sheet of paper and spend no more than 3 minutes seeing how much of the grid you can reproduce from memory.

It is an excellent self-assessment of what you have retained to that point from studying.
**Terminology Matching Exercise**
Every chapter has a terminology matching exercise to help reinforce your mastery of the agile lexicon that you will see on the ACP exam. It is also important because mastery of the “common language” enables successful interaction with other professionals in the field. The sooner you learn it, the sooner you will be on the road to success!

**Crossword Puzzles & Word Searches**
Also, you will find Crossword Puzzles and Word Searches at the end of each chapter. These exercises are designed to help reinforce key concepts and terminology in a fun and challenging manner.

**Chapter Practice Test & Answers**
At the end of every chapter, 20 questions were designed to test your command of that chapter’s specific material. These questions come from our database of over 1,000 questions and are based on chapter terms, formulas, calculations, and concepts. They will test both your knowledge and ability to handle various question formats, such as multiple-choice, fill in the blank, and application to scenarios – all of which you can expect to find on the actual certification exam.

The answers to the test questions are included for your review and edification. In the answer key, you will find explanations of both the right and the wrong answers. You can then use these answers to assess your understanding or interpretation of the subject that the question covers.

**Chapter End Notes**
1. Copyright © 2001 Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas; this declaration may be freely copied in any form, but only in its entirety through this notice.
2. See, for example, the work of Dr. Georgi Lozanov; and the research published by Dewey, Glasser, Hunter, Bloom, Goodlad, Gardner, Stallings, and others.
Introducing Agile Project Management

Chapter Highlights

In this Chapter we will begin with an overview of Agile Project Management and provide a quick, familiar example to help you create a high-level mental map of the subject. Then we will explore the idea of mapping Agile to the PMBOK® and review both agile planning and estimating as well as agile execution and control practices. With that mental map in place, we will cover the origins of Agile Project Management, the history of Lean systems thinking, and the application of Lean thinking to project management.

We will conclude by summarizing the agile frameworks and tools on PMI’s ACP Examination Content Outline, plus a few other frameworks, including:

- Scrum
- Extreme Programming (XP)
- Lean Software Development (LSD)
- Other Agile Frameworks
- Test Driven Development (TDD)
- Agile Modeling (AM)

Lastly, we will introduce the *Agile Project Management Processes Grid™*, a tool for ACP Exam preparation.
Overview of Agile Project Management

A Quick Familiar Example

Years of teaching experience have shown that sharing a quick, familiar example at the 50,000 foot level helps students create a mental map as they study Agile Project Management in greater depth. Below is an example of agile processes being applied in a situation that will be familiar to you.

You and three of your friends are hosting a dinner party together. Your project objective is to put on a successful party and remain friends afterwards! You are a cross-functional team because each of you has a different skill set. One is good at making cocktails, another is good at hors d’oeuvres, the third is good at entrees, and the fourth is good at desserts. Your team is not independently wealthy so it cannot simply outsource all of the work to a catering company. Consequently, you will be doing the work yourselves. To be successful, you want to establish mutual accountability as well as task integration so that there is support for each of you when it is time for the next deliverable.

The first agile principle in this example is that the team must have the necessary skills to complete the project. Agile is not a silver bullet! If a team does not have the required skills, even agile cannot help it successfully complete projects. The second agile principle here is that the team must be self-organized, highly-trusted, and accountable.

Because your team is also required on other projects (such as going to work) the team has agreed to use 4 iterations (in the evening) to complete the project. They are:

- Wednesday – Plan and acquire resources
- Thursday – Produce sub-components
- Friday – Complete and deliver the party
- Saturday – Clean up and do a retrospective before the next party

An iteration is simply a timebox within which work will be completed. In the agile world, work is done in iterations and a release can be the output of a single iteration or the output of several iterations that are inter-related by design choices.

Iteration #1 for your party is planning. Your team sits down and talks about what kinds of cocktails,
hors d’œuvres, entrées, and desserts you will serve. Based on the outcome of that discussion, the team prepares a shopping list of ingredients needing to be purchased in order to put on a successful dinner party. Each store where the team will shop is put on a separate piece of paper and assigned to a team member to purchase those items.

At the conclusion of iteration #1, the theme of the party, shopping lists, and a plan for its delivery have been established. These deliverables are referred to as a **potentially shippable product increment**. If a blizzard should blow into town on Thursday and prevent the party on Friday, the team has still produced a result, or output, which is useful when the project is resumed.

This agile principle is incremental building and frequent delivery of **potentially shippable product increments**. A potentially shippable product increment is anything that has value because the customer can see or use it to understand project progress. It may also have reference value for the team after an unexpected delay in order to restart the project. Also, a foreshadowing of the practice of *user stories* can be seen in the shopping lists. User stories are written documents that help the team understand what work needs to be done.

**Iteration #2** is preparation and logistics, specifically food preparation and set-up. Because you are working as a cross-functional team, you gather in the kitchen and help each other with washing, peeling, chopping, and storing the food ingredients you will use the following night.

At the end of iteration #2, you, once again, have an increment of value in a deliverable. All of the sub-components have been prepared or created and stored in the refrigerator.

If you should get an unexpected call that three of your guests missed their plane, forcing the party to be postponed until Saturday night, the iteration was still successful. The team produced a result, or output, which demonstrates project progress and is useful when the project is resumed. Furthermore, if the party was postponed for a week and some of the prep had to be redone due to spoilage, it would be similar to the work required to ramp a project back up after it has been stopped.

**The agile principles seen here are colocated work space (the kitchen) and also interrelated yet independent deliverables that demonstrate project progress.**

**Iteration #3** is execution; welcoming your guests and enjoying dinner with them. Everything goes as planned, good conversation occurs, and you enjoy your guests and the time around the table together. You have delivered (or consumed) another potentially shippable product increment.

In the *agile vocabulary*, we would refer to Friday’s output as both a **potentially shippable product increment** and as a *release*. It’s considered a release because it was the cumulative effort of the first 3 iterations and it was delivered to the customer – your friends. A key idea in Agile Project Management is that a release can be planned one of two ways. This example demonstrated one approach, where the release deadline was known and fixed and therefore the exact feature set was subject to some adjustment if needed. The focus was to deliver something the customer valued when it
was promised. For our example, the release was set for Friday and the customer (your friends) expected a deliverable of food and festivities, which they received.

The second approach to release planning is to define specifically what will be delivered and then analyze when it can be completed. After all, at a high level, the only way to fix both the scope and the date is to vary quality, and in agile, quality is never varied – it must be a working piece of functionality.

Iteration #4 is cleaning up with a retrospective meeting immediately afterwards. With agile in a normal environment – not the daily iterations we described here – the team holds two meetings at the end of each iteration. The first meeting is the review meeting where the potentially shippable product increment is presented to all interested stakeholders for their review and feedback. The review meeting is product focused. In our example, the customers ate the meal and were satisfied with the product, which was, in effect, a review meeting. The second meeting, which only the team attends, is the retrospective meeting. The retrospective meeting is process focused. The team uses that time to identify ways to improve how they create deliverables.

Because the project was completed by a cross-functional team, they also managed to remain friends, fulfilling a key objective. The project is complete, everyone is happy, and the experience with Agile Project Management has been both educational and successful.

This simplified example illustrates some of the key concepts and challenges of Agile Project Management. One of those key concepts is the idea of breaking up larger projects into interrelated, incremental deliverables. Those deliverables must be related and integrated in a fashion that continuously delivers value by building units of the solution. Each unit must be independent and build on prior work to move towards the final comprehensive solution. Finding the people in any organization who have the level of expertise needed to plan those types of increments is a very real challenge.

Is Agile Really Needed?

Even though there is an early precursor of agile in the concept of rolling wave planning, the last major tool recognized in the Project Management Institute’s “A Guide to the Project Management Body of Knowledge (PMBOK® Guide Second Edition)” was the Critical Chain in 1997. That fact raises the first question we should ask, “What has changed since then?” Consider the following:

- Google launched in September, 1998
- The iPod was unveiled in October, 2001
- The BlackBerry “smartphone” was released in January, 2002
- NASA’s Phoenix lander extracted Martian ice in June, 2007
- The iPad was introduced in April, 2010

Interestingly, the Apple iPad provides a “classic case study” in Agile Project Management. In Lean and agile terminology, it was a full function device that included the minimum marketable feature set, yet it was not a full feature tablet PC. Because it was focused on what the customer wanted, it sold 3 million devices in 80 days and almost 15 million devices in the 8 months of 2010, taking 75
percent market share of tablet PCs by the end of the year. That meant that it sold more units than all other tablet PCs combined.

The success of the iPad speaks eloquently to the success that agile enables. It also challenges organizational leaders who may feel an expectation for them to produce achievements like Steve Jobs.²

Even PMI acknowledged the increased demands and complexity of the project management universe when they moved beyond the long-cherished Iron Triangle – time, cost, scope – that was a part of every edition of the PMBOK through the Third Edition. With the release of the PMBOK, Fourth Edition, PMI took the traditional view of time, cost, scope, and added quality, risk, and customer satisfaction. The triangle became a hexagon in order to express the increased complexity that project managers now face in the everyday world. Soon project managers around the world will be speaking about the “Hell-of-a-Hexagon” that replaced the “Iron Triangle.” (See Figure 2.1.)

![Figure 2.1 Comparing the “Iron Triangle” to the New “Hell of a Hexagon.”](image)

There is an abundance of additional evidence that points to the added complexity faced by project managers. Consider the high project failure rates documented over the last couple of decades by the Standish Group in the aptly named CHAOS Reports.³ Or consider the report from Standish that proved only 20% of the features being delivered to users are in the “Always” or “Often Used” categories, while only 16% are “Sometimes Used,” and a full 64% fall into the “Rarely” and “Never Used” categories.

### Mapping Agile to the PMBOK®

Under the “Traditional” project management umbrella, PMI is the industry leader. PMI has the largest membership base, by far, of any professional user group for project managers and has developed the most recognized and best-respected credentials and certifications for practitioners in the project management field. There are, however, a host of smaller regional and local players that offer competing membership and certification choices. Regardless, PMI remains the leader and dominates the trends in identifying best practices because of its extensive research grants and educational scholarships.

Under the “Agile” project management umbrella, the Scrum Alliance (SA) is the biggest player because it has the largest membership base specifically in the agile sphere. It developed and controls the most recognized certification – the Certified Scrum Master (CSM) – for practitioners in the agile PM discipline. However, a host of smaller regional and local players offer competing memberships and certifications as well. (See Figure 2.2)
Because of the high profile that the SA has in the agile sphere, it is common to refer to Scrum and agile interchangeably - much like Kleenex® and facial tissue - but it is not always accurate to do so. Scrum is a variant, flavor, or approach within agile, however agile preceded Scrum and is broader than Scrum. Agile also includes several other notable frameworks, such as *Extreme Programming* (XP), *Lean Software Development* (LSD), and *Feature Driven Development* (FDD). Agile also has a group of others that make up a minute part of the market, including *Crystal, Dynamic Systems Development Method* (DSDM), *Agile Unified Process* (AUP), and *Spiral*. Finally, although the SA is the current leader in the agile space, PMI’s new ACP certification can be expected to challenge their position and eventually dominate the landscape over the next few years.

**Agile Planning and Estimating**

In order to compare traditional planning and estimating to agile, we have to first understand the assumptions that underpin each method.

In the traditional world of project management represented by the PMBOK®, the first assumption is that scope can and should be defined at the very beginning of the project (Figure 2.3). Although some evidence challenges the validity of assuming that scope can be accurately defined at the beginning of a project, it continues to be the starting assumption for traditional project management. PMs who have worked on large, complex projects have experienced change order process controls, change management boards, and any number of other tactics deployed to manage and control changes in scope. Despite all the effort to manage scope changes, it often proves futile. Nonetheless, traditional project management starts with an assumption that well-defined scope is possible.
This first assumption of well-defined scope drives the next assumption that desired dates and cost constraints can be honored. Once again, PMs who have worked on large complex projects often report the futility experienced when trying to get dates and costs to conform to a plan.

By comparison, the **first assumption for agile** methodologies is that the **customer clearly knows the date they wish to receive the solution and also the cost constraints that must be observed** (Figure 2.4). Because the customer knows these two pieces of data **firmly**, agile methodologies use dates and costs as the starting point for planning and estimating. Agile then proceeds based on the assumption that rigorous communication with the customer will drive value by refining project scope as development makes the options, and their costs, clearer.

Working with the customer, the **agile team strives to drive value by prioritizing the most important aspects of the project scope and developing them first.** In agile, planning and estimating is focused on creating accurate estimates that are reliable because they are within an appropriate, manageable, near-term time horizon. Beyond the near term horizon, agile estimating and planning focuses on avoiding the expensive illusion of false precision. Instead it uses tools, techniques, and tactics that provide robust, reliable planning at an intelligent, appropriate cost. The agile tools, techniques, and tactics used to do so will be covered in greater detail in subsequent chapters.

Before we move on, it is worthwhile to point out, as many traditional project manager’s would, that anyone who has worked on large complex projects knows that having an available, involved, and rational customer may be more elusive than trying to get dates and costs to conform to a plan that reflects changing external realities.

**Agile Execution and Control**

Execution and control in Agile Project Management relies on the use of **timeboxes** and **feedback cycles**. There are several types of timeboxes employed in Agile Project Management.

The highest level timebox is referred to as a **roadmap.** An agile roadmap is most equivalent to a program plan in the traditional project management world. Roadmaps are composed of release plans, the next lower level timebox in Agile Project Management.
The size of the timebox represented by a roadmap is the sum of the release plans within the roadmap (See Figure 2.5).

![Figure 2.5 Agile Roadmap of Release Plans.](image)

A release plan is a timebox equivalent to a project schedule in the traditional project management world. Release plans identify specific feature sets that represent a recognizable, logical component of the overall solution. Quite often, release plans represent the point at which deliverables can be used or implemented by customers. Release plans are composed of iteration plans. The size of the timebox represented by a release plan is the sum of the iterations within that release.

An iteration plan is the third timebox. Iteration plans are unique because they are a combination of a timebox and detailed work effort descriptions (See Figure 2.6). Each iteration contains the user stories, which describe the work effort for specific features or components that will be created by the agile team. Within iteration plans, user stories are decomposed into tasks, which can be estimated for the amount of work required to complete them. Iteration plans define the work that will be done in that specific timebox. Iteration plans are rolled up into release plans.
By definition, the size of the timebox for an iteration is stable. Based on organizational norms or rules, an iteration timebox will typically be defined as either two, three, or four weeks. Once the timebox for iterations has been defined, it remains fixed because rhythm helps the team increase speed, while stability helps the team improve quality over time.

Feedback cycles occur at several distinct points in the process.

One of those points is the daily meeting for the agile team. The daily meeting is sometimes referred to as a stand-up meeting or a Scrum meeting and is held to synchronize the activities of all of the team members. It also allows measurement of work progress against the iteration plan.

Another of those feedback points is the iteration review meeting, which occurs at the end of each iteration timebox and is product centric. At the end of each iteration, the agile team presents the completed deliverables to all interested stakeholders. This allows the stakeholders to see the most recent work product of the team and give feedback on how well it meets their needs and expectations. It provides transparency between the stakeholders’ needs and the agile team’s work, allowing adaptation to occur while it is easier and less expensive to make changes.

The third feedback point is the iteration retrospective meeting. The retrospective also occurs at the end of each iteration and is process centric. During the retrospective meeting, the agile team, Scrum Master, and Product Owner discuss process improvement ideas. It provides an opportunity for all members of the team to identify what changes would produce better work products, reduce errors, or improve communications.
The Origins of Agile Project Management

History of Lean Systems Thinking
Although arguments can be made that rigorous process thinking went into building the pyramids and many other significant accomplishments of antiquity, Henry Ford and his team of engineers – in particular Henry Gantt – are considered the first to truly integrate an entire production process. Around 1913 they integrated the ideas of interchangeable parts, standardized work units, and automated conveyors in order to produce what they described as production flow. The dramatic productivity gains made with a moving assembly line were revolutionary.

The revolution went beyond mere productivity gains and created insight into the theoretical foundations of mass-production manufacturing. Ford used process sequencing, single-purpose machines, and control-gate decision points to deliver required parts and sub-assemblies directly to the assembly line. Each of these ideas was a revolutionary break from prior standard practices of process grouping, general purpose machines, and batch-production decision points, which delivered parts still needing subassembly to the assembly line.

However, there were two significant problems with Ford’s approach. First, the production system could not accommodate variability and second, the manufacturing machines could not handle complexity. This was epitomized in Henry Ford’s, now infamous, statement that customers could have a Model T in any color they wanted…as long as they wanted black! But color wasn’t the only limitation. All Model T chassis had to be essentially identical even though customers could choose one of four body styles. Because the single purpose machines only worked on a single part, changes were not possible and the model cycle for the Model T ended up being longer than 15 years. In the end Ford’s system, while much more efficient than his competitors, lost out to other automakers who responded with many models and many options for each model.

The production systems of Ford’s competitors handled variability and complexity, but at the cost of much longer throughput times. Their larger, faster machines lowered costs per unit but continually increased throughput times and inventories. Compounding the problem, the lag time between process steps, because of the complex routing of parts, required immense management effort that eventually spawned the computerized Materials Requirements Planning (MRP) systems that have become common.

In the midst of this manufacturing milieu, Sakichi Toyoda, founder of Toyota, and his son Kiichiro Toyoda, worked to build upon what the Ford had done using ideas from W. Edwards Deming. Although they were unimpressed when they observed an American mass production assembly line, they were struck by an idea while shopping in a supermarket. They observed the simple idea where a customer took whatever soda they wanted and it was automatically replenished to await the next time a customer decided to take one.

As the Toyodas and Taiichi Ohno studied this situation, it occurred to them that a series of simple innovations might make it possible to provide both continuity in process flow and a wide variety in product offerings. The result was the Toyota Production System (TPS).

The revolutionary idea within the Toyota Production System was that the focus should be shifted from individual machines and their utilization, to optimizing the product flow through the whole manufacturing process. Using concepts that aligned parts and sub-assembly production to the actual volume needed, applied self-monitored quality, and integrated process sequencing with quick changeovers, TPS created a system where each step “requested” materials from the previous upstream step to meet current needs. The outcome was low cost, high variety, high quality, and very
short throughput times, which allowed them to respond to changing customer desires. As an added benefit, managing the immense MRP systems became much simpler and more accurate.

Today the ideas developed by W. Edwards Deming and the Toyota Production System are generally referred to as “Lean.” The concepts of Lean were first described by James Womack in the book *The Machine That Changed the World.* A few years later, Womack and Daniel T. Jones defined the five core Lean principles in their classic Lean Thinking.

**The Five Core Lean Principles are:**

1. Define the value the customer desires.
2. For each product, identify the value stream that provides customer value and challenge all of the wasted steps not directly providing it.
3. After removing the wasted steps, make the remaining value-added steps flow continuously through to the product.
4. Wherever possible, use “pulling” between steps to create continuous flow.
5. Continuously move toward perfection by reducing the number of steps, and the amount of time and information needed, to provide the customer value.

Because these five principles provide the theoretical foundation that influenced Agile Project Management, they are important to know and remember.

The terms pushing and pulling are Lean manufacturing concepts. **Pushing** signifies a “make to stock” (MTS) supply chain philosophy where production is not based on actual demand. Pulling is a “make to order” (MTO) approach where production is based on actual demand.

Lean thinking has spread through every industry, and nearly every country, causing leaders to adapt the tools and principles from manufacturing into services, healthcare, construction, and even charitable, institutional, and government settings. But Lean has only begun to influence senior managers and leaders compared to what the future will likely hold as time-to-market becomes a critical competitive differentiator.

The Lean principles summarized above have evolved into a set of core beliefs that should also be well understood in preparation for the ACP exam. Those core beliefs can be articulated as:

- The measure of success for any system or process is the amount of time between when ideas come in and when value is received by the customer.
- Any ad hoc system or process will produce unacceptable delay in customer value because it cannot be studied or improved upon. Therefore, processes must be defined in order to improve customer value.
- Most process errors are caused by the system, not the people who work in the system. Therefore, the people doing the work are the best qualified to define how to improve the system.
- The goal is to optimize the whole system, not merely improve individual steps. Therefore, optimizing the whole system or process by looking at when steps occur is a better path to improvement than trying to optimize the efficiency of each step.
- Because the goal is to optimize the whole system and because the people doing the work are the best qualified to improve the system, management must work with the team in order for the system to improve.
Teams, as well as systems, have inherent capacity limits that cannot be violated without subverting quality and sustainability. Therefore, teams are most efficient when the amount of work expected is within their capacity and efficiency is best improved by minimizing the amount of non-value or low-value work in process at any time.

These core beliefs create an agile paradigm where managers and teams work together toward the goal of maximum customer value. That fact is true whether the Lean principles are applied to software development, healthcare delivery, professional service delivery or any other field.

**Application of Lean Thinking to Project Management**

Proponents of traditional project management cite its success in the fields of engineering and construction as an indicator of its applicability to fields like software development. Since teams take requirements and build products that customers can use (not unlike engineering, construction, or other fields of product/service development and delivery), the theory implies traditional project management should work well. There are a number of problems with this thought process.

First, unlike construction where detailed blueprints are available before construction begins, or engineering where models or algorithms are available to specify specific processes before manufacturing begins, software development usually starts without clearly defined requirements or models that hold the rules for the complex variables that are involved.

Second, the immense variability and complexity of developing software makes the challenges faced by Ford pale in comparison. The variability is driven by the constraint of being human, which is to say we cannot perfectly perceive the best solution to a complex problem without going through incremental stages of development. The complexity is fueled by the wide range of variables, such as situational context and multiple platforms (i.e., PC, web, and mobile), which the solution must handle.

However, success in the software world – with all of its immense variability and complexity – has proven that Lean systems thinking, as embodied in Agile Project Management, can clarify what solution is needed and the process of discovery that can produce it. **Because the end goal is delivering value to a customer, Lean and agile processes are applicable to a great many fields outside software development.**

Applying the basic principles from Lean Manufacturing to project management requires the practitioner to accept the idea that fast, flexible flow in the development process – sometimes called the development pipeline – is possible. Many new practitioners have great skepticism about the whole idea that their specific industry could be modeled or managed as a fast, flexible pipeline. They discount that a value stream could be mapped or that mapping and refining it using Lean concepts as a guide would bear any useful benefits.

In order to help you prepare for the ACP exam, as well as prepare to apply Lean concepts in the workplace, we will now describe the Agile Project Management practices that are implied in Lean thinking. As we do this, you will see how agile practices have grown out of Lean principles. The reason for developing this understanding is twofold; first, when a question on the ACP exam describes a situation where an agile practitioner has found himself in a situation where standard agile practices won't work, you can use the Lean principles to guide you to the right answer and second, the development of this understanding will lead to better responses to workplace challenges.
Agile Project Management begins with the Lean concept that creating a sustainable stream of products requires directing business resources and focusing development teams so that results are based on prioritized business needs which are defined to create customer value. That, in turn, requires focusing on speeding up time-to-market by removing delays in the development process.

Another important concept implied by Lean, but made an explicit goal in Agile Project Management, is to improve communication. Quantitatively, improved communication reduces risk while also improving quality and dramatically increasing the likelihood of achieving real customer value.

Unfortunately, Agile Project Management practices tend to focus on communication at the local level – within the team, with the customer, and to a lesser degree, between multiple teams. Current Agile Project Management practices offer only limited support for improving communication across the enterprise or across the entire value stream. This weakness is being addressed by new practices in agile program and portfolio management.

Agile Project Management also embraces the Lean concepts of deferring commitment and eliminating waste as good ideas. Many practitioners are distracted by those word choices and fail to consider the evidence before making a judgment. But, as we mentioned earlier, when one considers the high project failure rates documented by the Standish Group in the CHAOS Reports and their research that showed 64 percent of the features being delivered to users fall into the “Rarely” or “Never Used” categories, the only viable conclusion is that when the Lean concepts of eliminating waste and deferring commitment are properly understood, they create value.

In Lean, and therefore also in Agile Project Management, deferring commitment means that decisions are made at the right time, sometimes referred to as the “last responsible moment.” Although this idea is counter-intuitive to many project managers because traditional project management has spent decades developing massive specifications at the beginning of a project, it is entirely sensible. Ask any experienced project manager whether it is better to (a) plan and estimate a project when very little is known about the problem or solution, or (b) plan and estimate a project when good information is known, and they will laugh at you because the answer is so obvious! Millions of change orders – change orders that could have been avoided – also validate the sensibility of this concept.

The concept is to resist making decisions too early, when needed information is not available, simply to create a sense of security or precision that will often turn out to be false. Conversely, the concept also warns against making decisions too late and incurring avoidable, higher costs, which usually occurs because the decision maker was too risk averse and wanted more accurate information used in the estimate.

The financial rationale for deferring commitments is quite straightforward. It is well known that with the application of additional effort – that is, time spent by resources that are usually expensive and in short supply – the accuracy of any estimate can be increased. As shown in Figure 2.7, 10 hours spent on an estimate might create 80% accuracy, while spending 20 hours might improve the accuracy to 90%. The problem is that a 12% improvement in accuracy (i.e. (90% - 80%)/80%), has required a 100% increase in cost, from 10 to 20 hours.

If that wasn’t onerous enough, consider the fact that the value of any estimate decreases with the passage of time. Estimates are most valuable early in the project when little is known, and least valuable late in the project when much is known. So the focus in Agile Project Management is to produce detailed estimates only when enough is known to get useful accuracy at a
reasonable cost. In Agile Project Management deferring commitment can be readily applied in defining requirements and doing analysis and estimating.

Figure 2.7 Accuracy Versus Cost of Estimate.

The goal of defining requirements and creating estimates should be to prioritize where resources are invested. If we simply stop to consider whether it is really necessary to define every single customer requirement, the answer should be a clear, “No.” What we need is great clarity about the requirements that will impact development resources on a reasonable time horizon – 60, 90, or 120 days.

Some requirements are more important, more urgent, or more technically complex than others. As a guide, Agile Project Management should start with those requirements that are the most important to the customer, involve safety, and create technical risk for things like scalability. Then, once those priorities are defined, move to requirements that improve marketability, performance, and flexibility. Finally, focus should shift to requirements that leverage opportunity or create comfort and luxury.

Recognize that commitments cost money because they spend time doing some kind of work, which can never be restored or the cost undone. Therefore, commitments should be spent doing work on the requirements that will bring the greatest value to the customer. Agile Project Management directs resources to the requirements that customers define as most important.

As we have shown, Agile Project Management embraces deferring commitment. Now, let’s see why the concept of eliminating waste is a good idea.

Most of the projects being managed today involve a considerable degree of architectural or technical risk, with risk being a good indicator of potential waste. After all, few things are as wasteful as building the wrong thing or building something no one will ever use. Therefore, eliminating waste has primary importance as a guideline for the Agile Project Management practitioner.

Waste comes in many forms. In software, it is code that is more complex than needed, causing undue defects and creating extra quality control work. In manufacturing, it is non-value-added work spent to create a product. In other contexts, waste is unneeded paperwork or documentation, or
missing paperwork or documentation that creates errors or rework, or a failure to create clarity that would have increased the speed at which the deliverable could have been created. Wherever waste is, Agile Project Management seeks ways to improve the system and eliminate it, because it is likely that errors will be repeated until the system that caused it is fixed.

Lean asserts that the most common and perhaps largest waste in traditional project management is the effort spent on detailed planning done too early in the project. To consider this idea from a Lean point of view, ask yourself, “When is it best to estimate, when little is known about the problem and solution domains or when much is known?” The answer is so obvious that when we ask that question in class, many students are hesitant to answer because they suspect it is a trick question.

Most project managers would acknowledge, especially on large, long projects, that accurate information is the least available during the early stages. Customers often have only a vague notion of how to describe the best solution. They often use the phrase, “I’ll know it when I see it,” which is referred to as the “IKIWIS syndrome” (pronounced icky-whizzy) to express their lack of clarity. Yet, this is when traditional project management often produces detailed requirements documents, very specific contract language, and detailed project plans.

Agile Project Management avoids this type of waste using a technique called emergent design. **Emergent design limits resource commitments and costs to those features that are currently necessary.** A comparison of the two concepts is shown in Figure 2.8, where a traditional waterfall approach that begins with a large effort to define everything and ever smaller efforts spent on elaboration over time, is contrasted with an agile approach, which only elaborates those things that are needed for reasonable clarity on a practical time horizon, such as 60, 90, or 120 days.

![Figure 2.8 Estimating Effort/Resources over Time.](image)

**Emergent design** has a dual meaning. It means that results emerge from an internal creative work process rather than being the result of an external blueprint. It also means that the design artifacts are more than the sum of the parts or a permutation of existing factors cobbled together by a team. It implies that the design process is creative and cannot be done by rote or by accident so it must be intentional.
When emergent design is applied in software development, it also integrates the discipline of **design patterns** to create application architecture that is durable and flexible, as well as automated acceptance and unit testing, to improve code clarity and reduce defects. **The use of design patterns enables code to more easily adapt to change by reducing complexity.** Reduced complexity is achieved by limiting coding to defined, current needs. Automated testing validates the design pattern, making the code safer to change should the need arise. Therefore, emergent design and automated testing combine to enable the deferral of commitments until implementation variables are reasonably understood.

A central tenet of Agile Project Management is that knowledge about the context of the problem and the variables of its solution are discovered and created as an integral part of the development process. **Agile Project Management solutions are created, built, or developed in stages so that as the customer uncovers specific real needs (remember IKIWISI?) the team can design and write code accordingly.** The analogy below may help clarify your understanding.

According to a recent article, at any given moment there are 500,000 passengers riding in airplanes over the United States. Due to factors like headwinds and crosswinds and collision avoidance routing, each of those flights is off course more than 90% of the time. The reason the vast number of those flights – or projects – are completed as intended is because the pilot uses accurate, transparent, real-time data to make minor course corrections as needed.

In Agile Project Management, the person who is the **voice of the customer**, called the **Product Owner** in the Scrum lexicon and **customer/proxy** in this book, is continuously taking in stakeholder feedback data and using it to provide the team with minor course corrections.

By doing Agile Project Management this way, the team delivers value quickly and avoids building things of little (or no) value. Remember that creating customer value is more of a discovery process than a building process. Software, buildings or medical devices have little inherent value. Value occurs when any of these enable the delivery of a product or service that solves a customer problem. Therefore, it may be more useful to think of product or service development – whether the deliverables are tangible or intangible – as a set of activities used to uncover the real needs, and the real problems, of customers and furthering the strategic goals of the organization by addressing them.

**Applying Lean to Agile Project Management implies accepting the mindset that it is necessary to deliver increments of the solution, early and often, so that the customer can experience specific aspects of the solution and reduce IKIWISI at each stage.**

Delivering increments early and often requires development to be done in iterations, which is referred to as **iterative development.**

The financial reason for doing iterative development is that customer value can be realized more quickly. Doing so improves market penetration, generates greater credibility for the business, creates strong customer loyalty, and increases profit margins. It also typically allows revenue streams to begin sooner, which, in turn, offsets the cost of subsequent development and reduces the total capital commitment required, directly increasing return on investment (ROI).

The financial benefits of Agile Project Management demonstrate the value of Lean’s focus on time and timing. Time is one of the core focuses in a Lean approach. Instead of the traditional project
management focus on resource utilization – which has driven matrix-type organizational structures and multitasking of workers - **Lean zeroes in on reducing the elapsed time from idea generation to delivery of value to the customer.** Of course, since time is money, when the team goes faster by using an improved system or process, costs go down.

Agile Project Management also uses the Lean focus on time and timing to reduce risk, in part, by eliminating delays that create waste. For example, some common delays that create waste in software are: requirements waiting to be verified as correct, work stoppages because a clarification is needed from a customer or analyst, and waiting for code that has been written, but needs to be tested. In healthcare, some common examples of waste are: delays experienced when a patient is waiting for insurance coverage to be verified as correct, when a pharmacy can’t dispense a prescription because a clarification is needed from the doctor, and when a pharmaceutical has passed testing, but is waiting to be approved by a regulator. These delays represent both risk and waste because delays increase the likelihood that something will be misunderstood and, in turn, multiply the potential of something going wrong.

By using iterative development steps, Agile Project Management creates the ability to make minor changes that move in the direction of the real solution without wasting effort. Borrowing from the Lean Manufacturing vocabulary, agile seeks to **minimize work-in-process (WIP).**

For Agile Project Management, WIP means those things that are described as 60% done or 80% done, or some other percent done in a traditional project status meeting. Because the customer cannot reduce IKIWISI and progress toward the real solution cannot be accurately measured, WIP has no value even though it has cost.

Whereas a traditional project may spend months or years going from 20% done to 30% done to 90% done, it isn’t until it is 100% done that the customer can truly ascertain if real value has been created. There are myriad examples of projects that accumulated astronomical costs while in a state of WIP only to be judged by the customer as having no value when the deliverables were finished. In those cases, the work done by the team has gone directly from WIP to waste, and despite elaborate risk management protocols, potential risk germinated into very real problems.

One well-known example of this was Motorola’s multibillion-dollar venture into satellite-based phones – the Iridium project. While some might argue that Iridium was a success from a technical perspective, despite remarkable financial forecasts and intense project management, the entire venture turned out to be a financial debacle because it failed to deliver customer value.

This could have been avoided by using Agile Project Management’s iterative development process, where the customer is given something at the end of each iteration that can be used, seen, applied or sampled, in order to produce clarity about needed course adjustments. The impact is organic risk mitigation and systematic value generation.
Summary of Agile Frameworks & Tools

As was previously stated, Lean provided a foundation upon which agile methodologies have developed. Likewise, agile provided a foundation, which is broader than Scrum, upon which the Scrum framework has developed into the most used framework with the largest association of practitioners.

A key point to note is that the lexicon used in this book is the most common taxonomy of methodologies, frameworks, and processes. (see figure 2.9) Methodologies provide the philosophical foundation for organizing frameworks, and in project management the two dominant choices are traditional, as embodied in the PMBOK®, and Agile. Methods are used to create and define frameworks as logical foundations. Frameworks are used to develop processes, which are the practical “how to” protocols used to guide things like sponsoring, organizing, funding, and controlling projects.

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Traditional</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework</td>
<td>PMBOK®</td>
<td>SCRUM</td>
</tr>
<tr>
<td>Process</td>
<td>Proprietary</td>
<td>Framework</td>
</tr>
</tbody>
</table>

Table 2.9 A Taxonomy of Project Management Methodologies, Frameworks, and Processes.

Figure 2.10 Agile Adoption by Framework.

Research indicates that the most common frameworks are Scrum, Scrum Hybrids, Extreme Programming (XP), Lean Software Development, and Feature Driven Development (FDD). These data are presented in Figure 2.10. There are also a host of other frameworks that each hold a very small market share, including Agile Unified Process (AUP), Crystal, Dynamic Systems Development Method (DSDM), and Spiral.
Each of the agile frameworks started in a specific context, which influenced how they applied Lean principles. In order to properly prepare for the ACP exam, and also to clarify which framework to start with when an organization first adopts Agile Project Management, we will briefly describe the various agile frameworks here.

Here, we summarize:

1. Scrum,
2. Extreme Programming
3. Lean Software Development
4. Other Frameworks
5. Test Driven Development
6. Agile Modeling (AM)

because PMI has specifically named them in its announcements for the ACP exam.

1. Scrum

(NOTE: As mentioned above, because Scrum is the dominant approach in Agile Project Management, we will develop an in-depth explanation of it in later chapters, but we will introduce and summarize it here.)

Scrum is a project management framework that uses iterative cycles and incremental deliverables to develop solutions. Scrum has been used extensively for agile software development, and more recently has begun to move into engineering, construction, and medical/pharmaceutical organizations.

The seeds of Scrum were planted in 1986 when Hirotaka Takeuchi and Ikujiro Nonaka wrote about a new approach to product development that increased speed and flexibility.9 They described a holistic or “rugby” approach of one cross-functional team moving through multiple overlapping phases, “passing the ball back and forth,” similar to what happens in a rugby Scrum.

However, the Scrum name first appeared in 1991 when Peter DeGrace and Leslie Stahl first referred to this as the “Scrum approach” in their book Wicked Problems, Righteous Solutions.10 But Jeff Sutherland, John Scumniotales and Jeff McKenna were the first to refer to it using the single word “Scrum” as the name of the approach they developed at the Easel Corporation.

In 1995, Jeff Sutherland and Ken Schwaber, another early luminary, collaborated in writing, sharing experiences and suggesting industry best practices. Their work became the first public presentation of what is now known as Scrum.

The Scrum framework is intended to be lightweight and contain a minimal set of practices and roles that can be customized after it has been learned and implemented.

In the Scrum framework, work is done during specified timeboxes, referred to as “sprints” (a synonym for iterations), which are typically between two and four weeks in length. Deliverables, referred to as “potentially shippable product” increments, are expected at the end of each sprint.

The ethos or heart of Scrum lies in the mutual commitments exchanged between the organization and the team. The organization agrees that the size of the timebox for all sprints, and the specific deliverables for each sprint, will be stable or frozen. In exchange for the organization’s promise to allow the team to focus and work uninterrupted, the team commits to delivering the specific deliverables at the end of the sprint regardless of unknown challenges that may surface while they
are in the process of creating the deliverables. In essence, the organization promises to be reliable, giving the team a stable work environment for the period of the sprint, and the team promises to be as creative as necessary to solve the problem and conquer the unknown.

**The main roles in Scrum are:**
- **Scrum Master (SM)** – The SM ensures the process is understood and followed, shielding the team from outside interference and removing impediments for the team.
- **Product Owner (PO)** – The PO is the “voice of the customer” representing the stakeholders and the business, and setting the priorities for deliverables.
- **Team** – The Team is a cross-functional group, which creates solutions by analyzing, designing, developing, testing, and implementing deliverables. It is assumed that the team has all the needed skills, is highly trusted, and self-managing.

Scrum supports self-organizing teams by preferring colocation of all team members so that immediate verbal communication is possible between all the subject matter experts on the team.

One **key principle in Scrum**, which is shared by the other frameworks, is that during a project, **customers need to be able to change their minds about requirements**, and the best way to meet those emerging needs is a combination of short term stability (during the sprint) and long term flexibility (outside the sprint). Scrum and other agile frameworks accept that complex problems cannot be fully defined in advance; they use the scientifically validated empirical process control approach – transparency, inspection, and adaptation – to focus the team on quickly delivering results that move through the cone of uncertainty towards a solution in the midst of emerging requirements.

Again, because Scrum and Scrum Hybrids are the framework being used by 75% of organizations implementing agile, it will be highlighted throughout this book. At this point, however, we will move on to summarizing the other frameworks.

**2. Extreme Programming (XP)**

Second to Scrum, in adoption rate and marketplace usage, is Extreme Programming (XP).

XP began as a response to two major changes in software development. First, new object-oriented programming methods began to replace procedural programming as the approach favored by businesses. Second, speed-to-market became a strategic growth issue as the Internet and the dot-com boom realigned the competitive business landscape. Shorter product life-cycles demanded agility to handle rapidly changing customer requirements, creating an environment that was incompatible with traditional software development.

XP was intended to respond to these pressures and optimize business value. Because full implementation of XP requires a very mature operating environment and also very high levels of discipline, it is only well-suited to a limited number of organizations.

The ideas for Extreme Programming were formalized by Kent Beck in his book on the method, *Extreme Programming Explained.* As its name implies, the context in which XP started was software, so it is a programmer-centric framework focused on technical practices that promote skillful software development. Also as its name implies, XP applies techniques and practices with extreme rigor as long as doing so improves results.

Beginning in 1999, a very elementary “XP System Map,” as shown in Figure 2.11, has been used to illustrate the core XP concepts.
Over the course of time, this has expanded with best practices emerging, such as continuous code reviews, pair programming, unit testing, modularity, bottom-up design, incremental design patterns and refactoring of all code.

XP currently defines five basic activities within the software development process:

- **Coding** - As one might expect given the context of its start, XP states that the most important product of the development process is software code because without code, there is no working product.
- **Testing** - Testing, and predictably lots of it. Since the most valuable product is code, testing has extreme value because the more flaws it can eliminate, the more working code (and therefore value) can be delivered.
- **Listening** - Listening, although common sense, is a departure from the expected “short list” for the stereotype of “propeller heads.” In XP, programmers listen to what “business logic” the customer needs so they can understand those needs well enough to discern the technical aspects of a viable solution, unless it is discovered that the problem cannot be solved.
- **Designing** - Designing, specifically for simplicity. The goal is to limit system complexity at the threshold where it is mandatory, and no more than that, and prevent dependencies becoming unclear. XP seeks to avoid that trap by creating modular designs that organize the system logic with the minimum amount of dependencies.
- **Communicating** - Communication is, again, unexpected for the stereotype. Whereas listening defined programmer behavior towards customers, communicating to the customer expresses the need to have system requirements from the customer for the developers.

The XP goal is to rapidly create and distribute organizational knowledge to the development team, creating a unified view with the customers and users of the system.

For the ACP exam, students should note that the roles on an XP project include:

- **Customer**: The customer creates and prioritizes the stories to be developed. Unlike other agile frameworks, the customer can vary the release date by adding or removing stories from the backlog to be delivered in any given release. Similar to other frameworks, the customer sets the development priorities.
• **Programmer:** The programmers estimate stories, accept responsibility for tasks, write tests, and write code, usually working in pairs.

• **Coach:** The coach, while preferred, is optional and monitors the process, mentors the team on XP processes and techniques, and helps the team identify and focus on risks and optimization opportunities.

• **Tracker:** The tracker, another preferred, but optional position, monitors team progress and warns when redistributing tasks might be required to adjust the schedule. Sometimes a programmer “doubles” as tracker for the same pair, and sometimes a programmer serves as tracker for a different pair.

The following are considered the 12 Core Practices of XP:

- **Planning Game:** The Planning Game is a technique used to elicit new requirements from the customer, with the team then giving an estimate of the effort required to develop and implement it.

- **Small releases:** Begin with the smallest useful feature set. Release early and often, adding a few features each time.

- **Product theme:** Each project has an underlying theme or metaphor, which provides an easily remembered naming convention.

- **Simple design:** Employ the simplest possible design to get the job done.

- **Test-driven development (TDD):** Write a test before adding a feature. When the suite runs, the job is done. We will discuss TDD in more detail later.

- **Refactoring:** Refactoring refers to constantly improving a product’s internal design (e.g., rewriting code) without changing its behavior to make the product more reliable and adaptable.

- **Pair programming:** Pair programming is a technique where two programmers work together on one computer. The first programmer types in code while the second programmer reviews each line of code as it is written. The first programmer is called the driver. The second programmer is called the observer or navigator. They switch roles frequently. The observer considers things like potential problems with strategic choices, as well as ideas for improvements. The driver focuses all of his or her attention on tactical choices and completing the current programming task.

- **Collective code ownership:** This is a practice where everyone shares responsibility for the quality of the code, allowing anyone to make necessary changes anywhere.

- **Continuous integration:** With continuous integration, all product changes (i.e., to the code base) are integrated at least daily. Unit tests have to run at 100% both pre and post-integration.

- **Forty hour work week:** The development team works a standard 40 hour week. A maximum of one week of overtime is allowed. If additional overtime is needed, this is considered a serious failure of the agile process and must be evaluated.

- **On site customer:** The development team has continuous access to an on site customer. For commercial software with many stakeholders, a proxy may be used.

- **Coding standards:** Everyone codes to the same standards and conventions.

Agile, in general, has a goal of using face-to-face (F2F) and osmotic communication. This is typically achieved through colocation arrangements described as a “caves and commons” area.

The team is located together where they have a “commons” area everyone shares to facilitate instant or immediate communication. The co-locate area also has “caves,” which provide private space for
when team members need to retreat and focus in order to solve a challenging issue. This type of arrangement fosters osmotic communication; team members regularly pick up important data “by osmosis” – data which they otherwise might not get.

Osmotic communication can be described as a team member picking up pieces of a conversation in a common area and then being able to make a meaningful contribution even though he or she was not fully engaged in the discussion.

Another view on the term osmotic communication draws on science where an element moves through a semi-permeable membrane into another organism. In agile, the experience of team learning has been described as an almost effortless or unconscious assimilation or diffusion of knowledge that seemed “osmotic.” The explanation seems to be that the mind becomes semi-permeable when it is engaged in solving a problem. So simply being immersed in an environment where team discussions are happening around the person whose mind is so engaged, causes information to flow in or act upon the mind.

Among the common criticisms of XP are problems with requirements stability, lack of documented design specifications at the architectural or systems level, and lack of a document trail defining compromises made to solve user conflicts. There is also controversy about the “reality” of XP practices such as the demand to have the customer/proxy on-site, or the requirement to have a unified customer viewpoint versus a single programming organization. Because scalability is often required in today’s globally distributed world, XP has been perceived as having limited value by many organizations.

3. Lean Software Development

As with XP, the name of the Lean Software Development (LSD) framework gives away the context of its start up. The goal of the LSD framework is to eliminate any unnecessary burden or overhead from the software development process. Unlike other agile development frameworks, LSD is concerned with how the organization perceives and articulates the systems they want, and that concern shows up during requirements gathering and documentation efforts.

The original LSD framework descends from the work of W. Edward Deming, with more recent improvements being drawn from the Theory of Constraints. Lean manufacturing – and hence the name, Lean Software Development – grew out of Deming’s work on Total Quality Management (TQM). TQM emphasized the importance of process and the absolute truth that the people using the process are the ones who must build and improve it. TQM is very quantitative and metric centric, so it is no surprise that metrics play a significant role in LSD as well.

The newer additions to LSD come from the work of Dr. Eliyahu M. Goldratt on the Theory of Constraints (TOC). The TOC stressed the need to first identify constraints and then remove, or at least improve the constraints, in order to improve the business organization, which is a self-contained system.

Therefore LSD processes pursue the goal of defining, developing, and delivering complex software systems that exactly meet the competitive business challenge. The emphasis on the project management aspects of software development, rather than the technical ones, means LSD shares thinking with Scrum. Both frameworks seek to manage costs and improve the project’s ROI.

Because of LSD’s focus on gathering the correct requirements (those with the biggest impact to the business) and defining them with great clarity and completeness so they can be verified, the customer plays a central, critical role. Continuous customer feedback regarding the functional requirements needed to create business value, also helps identify missing, conflicting, and incomplete requirements.
More so than many other agile frameworks, LSD uses the focus on requirements to create an abundance of quantitative metrics to guide configuration and management decisions so that project failure can be avoided. Because LSD focuses on the "root cause" of issues such as resource constraints, teams lacking the correct skills, and excessive team membership churn, it often creates political discomfort. That discomfort will be most pronounced in organizations where projects are executed within departmental silos, or other insulated settings.

Unlike other frameworks that have semi-strict role definitions, LSD uses a more cross-functional approach. Team members are cross-trained on functional and technical facets of the system, as well as on the business problems to be solved or business value the system features are expected to provide to the customer.

Because LSD focuses more on the project management side and requires few specific technical practices, it can be integrated rather easily with other agile frameworks such as XP, which focuses on the technical facets of software development.

The LSD framework upholds seven principles, which began in Lean manufacturing and have been extended in its approach to project management. The principles are:

1. **Eliminate Waste** - It does not add customer value so eliminate it!
2. **Build Quality In** - Validate all assumptions and use metrics throughout the process to ensure practices create value, otherwise discard them.
3. **Create Knowledge** - Use short iterative cycles to get continuous feedback and ensure focus is on the right things.
4. **Defer Commitment** - Don’t make decisions until a clear understanding of the problem, the solution choices, and the tradeoffs of each are available.
5. **Deliver Fast** - Identify business issues as quickly as possible and then deliver a system or feature that solves them.
6. **Respect People** - Only the employees using the system can improve it, so empower the team to succeed by enabling them to improve it.
7. **Optimize the Whole** - Always use cross-functional teams so important or critical facets of the problem aren’t overlooked and the solution design will solve it.

### 4. Other Frameworks

While these frameworks are not specifically mentioned by PMI as part of the knowledge base required for the ACP exam, probably because they hold such a small market share, it is nonetheless worthwhile to have a cursory understanding of Feature Driven Development (FDD), Agile Unified Process (AUP), and Crystal.

The other frameworks – Dynamic Systems Development Method (DSDM), Essential Unified Process (EssUP), Open Unified Process (OpenUP), and Spiral – make up such a minute part of the market that we will not cover them.

**Feature Driven Development (FDD)**

Among the various agile frameworks, Feature Driven Development (FDD) may be the most “tightly wrapped.” Whereas other frameworks begin with a set of principles or processes, the FDD core is the domain model.
Defining the domain model is a mandatory first step for FDD. Creating it requires that domain knowledge be collected from subject matter experts (SMEs), referred to as Domain Experts, and then integrated into a cohesive model that accurately represents the problem domain. Using the Domain Model, validated requirements are analyzed and a plan is drawn up to create the solution and determine the resources needed to build it.

Typically requirements are gathered using a top-down approach where general business practices are defined as Subject Areas. Subject Areas are decomposed into business activities that are defined as Feature Sets. Feature Sets are decomposed into tasks, defined as Features, that can be clearly specified and accurately estimated.

The development activities of building the features drive the team's work (hence the name, Feature Driven Development) through iterations that are typically no more than two weeks long. Sets of features are developed, one after the other in successive iterations. For reporting purposes, data is collected at the feature level and rolled up into Feature Sets, and Feature Sets can then be rolled up into Subject Areas.

As shown in Figure 2.12, the FDD framework requires five steps that must be followed in order to create a very specific process. The steps are:

1. Develop Overall (Domain) Model
2. Build a Feature List
3. Plan by Feature
4. Design by Feature
5. Build by Feature

In the FDD framework everything is planned, designed, built, and managed at the feature level. Other levels like feature sets, domain areas, and requirements lists are available for higher-level planning and reporting, but the “master key” is the feature.

In FDD a feature is defined as a small, client-valued function. It is expressed as an action that causes a result to an object. The formula for defining features is written in the
form: <action> <result> [of|to|for|from] <object>, where the information in the square brackets is optional and used to make the feature statement easier to read. For example, a feature statement might read, “Calculate monthly payment for car loan.” The verb "calculate" is the <action>, the noun "monthly payment" is the <result>, and the noun "car loan" is the <object>.

FDD also defines more specific roles and responsibilities than most other agile frameworks. The nine roles are:

1. **Project Manager** – The PM is responsible for all administrative, financial, and reporting facets of the project.
2. **Chief Architect** – The CA controls the design of the Domain Model and solution (system or process) and manages the technical architecture, design sessions, and code reviews.
3. **Development Manager** – The DM manages daily development activities, coordinates the development team, and resolves resource issues.
4. **Chief Programmer** – The CP is a senior developer who is responsible for a specific Feature Set and manages their design and development activities.
5. **Class Owner** – The CO is a developer who reports to the CP and designs, codes, tests, and documents features as they are implemented.
6. **Domain Expert** – The DE is a business SME and stakeholder who defines requirements as features that the solution must provide. Business analysts are the most common DEs, but anyone with knowledge of how the system will impact the business can be a DE.
7. **Tester** – The Tester is responsible for validating that features perform as defined.
8. **Deployer** – The Deployer manages the data definitions and conversions and supports the deployment of code to the various platforms.
9. **Technical Writer** – The TW creates and maintains the documentation that users will need to operate the system.

Because of the number and specificity of the roles in FDD, it does not use the principle of **shared ownership of code** and **artifacts** the way other agile frameworks do.

For the ACP exam, students want to know the definitions of shared ownership of code and artifacts:

**Shared ownership of code**, also known as collective code ownership, means every programmer is allowed to change any part of the code. Therefore, every programmer is responsible for all the code. Pair programming supports this idea because as programmers work in various pairs, they get exposed to all the code in the system. Advocates say this is a major advantage because it speeds up development and any errors in the code are fixed by programmers when they encounter them. The obvious risk comes from programmers who do not see subtle dependencies. Well-defined unit tests are used to address this issue.

Artifacts are the data records that document a record of the projects inputs, outputs, and progress points. In Scrum, the four principal artifacts are the Product Backlog, Release Burn-down, Iteration Backlog, and Iteration Burn-down. Artifacts may be kept electronically, but are intended to be used as visible information displays.

Another unique aspect of FDD is the very specific report style used to report project progress. Most agile frameworks use lists to track requirements and work done. In FDD, all lists correlate to specific features while also using a Feature Set Progress Report. The Feature Set Progress Report tracks feature development progress by subject area using specific color codes and percentage complete. Light green means the Feature Set is on schedule, dark green means the Feature Set has been
completed, and grey means the Feature Set is behind schedule. Feature progress is rolled up into a table showing progress against Feature Set milestones in the Feature Set Progress Report.

Features are completed a line graph showing the project’s total progress by day or week. The exact state of each feature is documented in a table with six specific milestones. They are:

1. Domain Walkthrough
2. Design
3. Design Inspection
4. Code
5. Code Inspection
6. Promote to Build

From these steps, stakeholders can see how the project is progressing.

**Agile Unified Process (AUP)**

The Agile Unified Process (AUP) framework is a simplified version of the Rational Unified Process (RUP). The RUP is an iterative software development framework created by the Rational Software Corporation, a division of IBM, and developed by Scott Ambler and Larry Constantine. It is generally considered a “high ceremony” framework because it specifies many activities and artifacts for each project. The AUP defined an easier-to-use approach for developing software based on integration of agile techniques and concepts. The AUP includes test driven development (TDD), agile modeling, agile change management, and code refactoring techniques to improve productivity.

The AUP is based on six philosophies, which are:

1. **Competence** – The team knows what it’s doing. They won’t read detailed process documentation, instead will apply high-level guidance and standards.
2. **Simplicity** – Describe things concisely on a few pages, not reams of pages.
3. **Agility** – The AUP conforms to the values and principles of the Agile Alliance.
4. **Activity** – Focus on only the high-value activities that count. Ignore the noise.
5. **Tools** – Simple tools are often the best, so AUP is independent of any toolset. AUP recommends using the tools best suited for the job.
6. **Tailor** – AUP works best when tailored to the needs defined by the context.

The AUP kept the four major phases of the UP, which are:

1. **Inception** – Inception cultivates a shared understanding of the project scope and defines architectural choices.
2. **Elaboration** – Elaboration develops the understanding of the system into requirements and validates architectural choices.
3. **Construction** – Construction occurs until system development is completed.
4. **Transition** – Transition all testing and system deployment to production.

These phases occur serially and conclude after a specified milestone is accomplished. The AUP made a significant change to the UP when it defined two types of iterations. The first is a development release iteration that reaches the Transition milestone when it is deployed to a non-production environment, such as a quality-assurance and testing environment. The second is a production release iteration that reaches the Transition milestone when it is deployed to the production environment.
The AUP simplified the RUP by focusing on seven disciplines, which are:

1. **Model** – Use a model to represent the organization’s business approach, the problem domain, and any viable solution to solve the problem.
2. **Implement** – Code the model(s) into executable code and perform unit testing.
3. **Test** – Apply additional tests to find defects, validate the system design works, verify the requirements are satisfied, and ensure code quality.
4. **Deploy** – Plan and deliver the system for end users.
5. **Configuration Management** – Control all project artifacts, including version tracking and change management.
6. **Project Management** – Provide project management, including scope, resource, risk and progress management, and coordination of external interfaces, to achieve an on time, on budget completion.
7. **Environment** – Provide process guidance standards and ensure needed tools are available for the team.

Risk management in AUP prefers that high-risk elements be prioritized early in development. For example, the core architecture is developed during the Elaboration phase to validate requirements, assumptions, and address technical risks. AUP also uses a risk list to document and manage risks throughout the development process.

**Crystal**

Among the agile frameworks, Crystal is the only one that is actually a family of frameworks that vary based on project size and criticality. Crystal was named by the author, Alistair Cockburn, in his book *Crystal Clear: A Human-Powered Methodology for Small Teams*. Each specific framework within the family is designated by a color that correlates to the hardness of a geological crystal, thereby implying the project's size and criticality.

In Crystal, size is defined by the number of people involved in a project. As team size grows, more formality or ceremony is added to the structure, artifacts, and management. In Crystal, criticality is defined as the potential damage the system can cause. For example, an operating room system malfunction can cause far more damage than a video game controller that misfires. As project criticality increases, the formality of control needs to increase to guarantee that requirements are fulfilled.

Cockburn once stated that even though the Crystal frameworks share elements, they are not expected to be upward or downward compatible. For example, a Crystal Clear project does not transition to a Crystal Maroon project because of an increase in size and criticality. Instead should the project become Maroon, the project should adopt the Maroon framework, not expand the prior Crystal Clear practices.

This approach offsets a significant weakness in Crystal. The only authoritatively published specific information for applying Crystal is for the Clear and Orange levels, with some discussion of a variant called Crystal Orange Web, surfacing recently. Cockburn has dismissed this criticism saying, in effect, if you feel you need details you don’t have adequate background to use Crystal. We find that response less than satisfying. Nonetheless, Crystal does encapsulate some insightful thinking.

Regardless of which Crystal version is chosen, they all share seven key principles. The seven key principles are:

1. **Frequent Delivery** – FD means stakeholders and customers receive deliverables every couple of months, at a minimum. Large deliverables may not go into the production
environment, but intermediate versions of them should be available in a test environment where stakeholders and customers can provide feedback.

2. **Continual Feedback** – CF means the team meets on a regular basis to discuss project activities with stakeholders and uses feedback to confirm that the project is headed in the desired direction or make adjustments as needed.

3. **Constant Communication** – CC means that on small projects the team is collocated in the same room, while larger projects are collocated in the same facility or on the same campus. CC also means that teams have frequent access to the persons defining requirements.

4. **Safety** – Safety is recognized in Crystal in two forms. The first form, which is common to most agile frameworks, recognizes the need for a “safe zone” where team members can communicate without fear of reprisal. This allows hard truths that might impact the overall process to be brought to light early, because team effectiveness suffers without access to the truth during the project. The second form, which is unique to Crystal, recognizes that some software projects affect end-user safety. For example, a weapons system is far more critical than a music organizer.

5. **Focus** – The top two or three priorities are clearly shared with the team and the team is given uninterrupted time to complete them.

6. **Access** – As with most agile frameworks, the team must have adequate access to end users of the system while it is being built.

7. **Automated Tests and Integration** – Testing and integration must be supported by automated versioning, testing, and integration of system components.

Determining the Crystal framework to use for a project is done on a grid like Figure 2.13. The horizontal, X-axis, is divided into five columns defined by colors correlated to size, ranging from Clear to Maroon. The numbers in the cells are the upper size limit of the project team. As the project team size increases, the framework choice moves from left to the right, from Clear to Maroon, because the project is expected to be harder.
The vertical, Y-axis, is divided into four rows, correlated to criticality, which rise from the bottom upward, and noted as “C” for Comfort issues, “D” for Discretionary Money issues, “E” for Essential Money issues, and “L” for Life impacting issues. As the criticality of a project increases the framework choice moves from the bottom to the top of the column because the framework must adapt to the additional requirements and artifacts needed.

In the Crystal Clear framework, roles are limited and all team members are expected to be colocated in the same room. The most important role is the Senior Designer who makes all the technical decisions. The team decides what artifacts will be produced and what coding standards and test practices will be followed because project milestones are actual working software, not documents. Other duties such as project management, business analysis, and testing are shared by the Sponsor and Programmer(s). Working software is expected to be released every 60 to 90 days, even if work is performed in shorter iterations.

Nothing has been published regarding the Crystal Yellow framework.

In the Crystal Orange framework the number of roles increases, and can vary between organizations and projects, but typically include the Sponsor, Project Manager, Business Analyst, Architect, Designer(s), Programmer(s), QA Engineer, and Tester(s). There is greater formality about what artifacts will be produced and what coding standards and test practices will be followed. Working software is expected to be released every 90 to 120 days, even if work is performed in shorter iterations.

Crystal Orange defines the artifacts and deliverables as:

- Requirements Document
- Release Sequence (Schedule)
- Project Schedule
- Status Reports
- User Interface Design Document (if a UI is delivered)
- Object Model(s)
- User Documentation or Manual
- Test Cases

Nothing has been published regarding the Crystal Red and Maroon frameworks.

5. Test Driven Development (TDD)

Test-driven development (TDD) is a software development process; an Agile Project Management tool, not a framework. It is used in conjunction with a framework.

TDD has grown out of the test-first concepts of extreme programming, and more recently has enjoyed general interest on its own. The most common usage occurs when automating an existing manual process that has a current testing process. It is also applied to improve legacy code developed with older languages.

TDD uses repetition of very short development cycles where a developer writes test preconditions, test controls, and test reporting based on predicted outcomes that define a desired improvement or new function. The developer then writes code and uses automated testing software tools to monitor the execution of the code to test the actual outcomes against the predicted outcomes. If the code fails the automated test case, the developer refines it until code is produced.
that passes. As a final step the code is refactored, processed by a tool that makes small changes in the code that do not modify its functionality, but improves attributes like readability, complexity, maintainability, and compliance with architecture or object model standards for extensibility – until the new code meets established standards.

In TDD, the programmer focuses on passing the immediate test and only then considers how to handle exceptions, errors, and rare circumstances. It is easy to jump to the conclusion that TDD only provides validation of the correctness of code, but it is useful to recognize that it can also drive program design. The discipline of understanding how customers will use specific functionality, which is needed to create the test cases, raises awareness and concern about the interface before implementation occurs and drives design in a beneficial direction.

However, TDD can be difficult to apply with programs that integrate to databases, where full functional tests are required to determine success or failure or where code is dependent on specific network configurations. It also has a possibly significant weakness because tests are created by the developer who is also solving them. Therefore tests may miss the developer’s “blind spots” regarding user specifications. And, of course, management may object that the tests increase overhead and maintenance.

6. Agile Modeling (AM)

Agile Modeling (AM) is also not a framework, but a modeling tool for software development projects. It is used in conjunction with a framework.

For the ACP exam, students should note that AM is a collection of values, principles, and practices used to create models for software development projects. It is intended to be an effective, light-weight approach that is tailored into a framework such as Scrum, XP, or Lean. AM creates effective models that support development of solutions using a performance-based approach.

The basic AM principle is to create multiple models in small increments because any given model is bound to include some inaccuracies. The practice is to create an abstract representation of the software then prove or disprove its performance with code that actually works or does not work. Using the right artifacts from each model, the team improves its understanding of the situation, and iterates to the next model and artifacts, to follow a continuous forward march to a usable solution. Active stakeholder participation in AM is critical because the project stakeholders know what the result of a successful model will be and can provide the feedback needed to improve between each one.

The principle of applied simplicity is used in AM to focus on the practice of only creating models for the current facet of the problem and avoiding large, detailed models. The principle of open communication is practiced in AM by displaying models on walls or Wiki’s, embracing collective ownership of artifacts, following modeling standards, and using group-based model development.

AM suggests doing just enough modeling to understand the scope of the problem and the architecture of possible solutions, then using development iterations to improve those models until a solution can be created.
The Agile Process Map™ & Agile PM Processes Grid™

Agile Process Map™
Now that you have been exposed to a significant part of the agile lexicon, it is time to put those ideas together in an Agile Process Map™ that can be used to guide and integrate the detailed learning in the rest of this course.

As you may have gathered from what you have learned so far, the agile worldview centers around the team’s work as the point of value creation. So from the team-centric view, the macro perspective is a process that moves from a “steady state” to a “transition state” to a “steady state.”

The first steady state encompasses the activities of the customer/proxy, or in Scrum vernacular the Product Owner (PO), who is receiving, analyzing, and prioritizing the features required for a successful solution. That work is kept in an artifact referred to in Scrum as the product backlog. The Product Backlog is equivalent to the product specification or requirements list in traditional project management. It is, however, significantly different because the PO is continuously grooming it based on information being received from internal and external sources. As priorities change, system features can be promoted or demoted. As the project moves forward, features that were on the future horizon enter the current horizon and are analyzed and estimated. So the Product Backlog is in a state of flux, but from the team’s perspective it is in a steady state. That is because the team interacts with the Product Backlog only at the beginning of each iteration in order to negotiate with the PO and decide which features will be included in the next iteration. Once those features are agreed upon and fully committed to, they cannot be changed. This is shown in Figure 2.14.

So for the PO, the backlog is in a state of transition, but for the team that flux is outside their concern. Once the team has fully committed to the Iteration Backlog, which is the portion of the Product Backlog being developed in the current iteration, enters a state of transition. The team has committed to doing whatever is necessary to change the current state of those features into the future state as the goal of that iteration. At this point, conversely, for the PO the iteration (or in Scrum vernacular sprint) backlog is in a steady state and may not be changed.
The desired future state of those features is, by definition, the potentially shippable product increment, which is the goal of that iteration. The team will focus all of its energy on building that part of the solution or system and at the end of the iteration, will demonstrate for all interested stakeholders what has been created at a review meeting. The review meeting is a product-centric meeting where acceptance of the deliverables and considerations for future enhancements is discussed. As far as the team is concerned, once they have demonstrated that the features work as agreed upon at the beginning of the iteration, the second steady state has been achieved. This is illustrated in Figure 2.15.

Between those two steady states is the state of transition, which is where the agile team lives its life. In the state of transition, there are only two constants - the duration and the goal of the iteration. Everything else is in a constant state of change, driving the need for the team’s daily meeting to synchronize and plan. Each day when the team meets, each member briefly explains what they have done since the last meeting, what they will do before the next meeting, and any impediments interfering with their ability to be effective and productive. The team will use that information to self-manage. They will re-plan as needed, synchronize hand-offs, rally to support one another, and also hold one another accountable. Each day, each member is expected to make reasonable progress towards the fully committed, agreed-upon iteration goal. As needed, the team will also meet with the PO to clarify questions or concerns about elements, components or behaviors of the system in order to make sure it will meet the definition of done that the PO supplied at the beginning. Sometimes discoveries or insights will come out of the daily meeting. Those discoveries and insights are forwarded to the PO for use in grooming the Product Backlog.

By having daily results that are measurable, external control is unnecessary because internal self-discipline has been created.

The final part of the process is the retrospective meeting. The retrospective meeting is attended by the team - and sometimes the PO - only. It is a process-centric meeting where the team identifies how it can improve its process of creating potentially shippable product increments. Typically, the review meeting and the retrospective meeting are the first and second halves of a single meeting for the team.
Chapter End Notes


2 Steven Paul "Steve" Jobs (February 24, 1955 – October 5, 2011) was a visionary widely recognized as a charismatic pioneer of the personal computer revolution. He was co-founder, chairman, and chief executive officer of Apple Inc.


4 William Edwards Deming (October 14, 1900 – December 20, 1993) is perhaps best known for his work in Japan. He taught how to improve product quality through the application of statistical methods. Deming made a significant contribution to Japan’s later reputation for innovative high-quality products. Despite being a hero in Japan, he was only beginning to be recognized in the U.S. at the time of his death.

5 Taiichi Ohno (February 29, 1912 - May 28, 1990) is considered to be the father of the Toyota Production System, which became Lean Manufacturing in the U.S.


8 *5th Annual State of Agile Development Survey Results* (©qaSignature, Inc. 2011): This fifth annual survey was conducted between August 11 and October 31, 2010. It includes information from 4,770 participants from 91 countries. Data was analyzed and prepared into a summary report by Analysis.Net Research, an independent survey consultancy.


12 http://www.extremeprogramming.org


Initiating Projects

Chapter Highlights

In this chapter, we will build on the overview from the prior chapter. We will have an in-depth discussion of Initiate, the first process in the Agile Project Management Processes Grid™. We will describe how to identify stakeholders and build their engagement, how to create value-driven deliveries using adaptive planning, how to enable team performance and clarify risk, and how to support effective communication and continuous improvement.
Overview of Agile Project Initiation

The process of initiating projects in an agile framework is just as important as it is in a traditional framework, if not more so. It is, however, quite different because of its focus on prioritized, cyclical, iterative development. As you will recall from Chapter 2, when we compared the traditional and agile approaches to estimating, as shown in Figure 2.8, the traditional approach started with a large effort to elaborate everything in detail. The agile approach meanwhile, elaborated in detail only those things that had been prioritized and were on a practical time horizon, such as 60, 90, or 120 days, noting everything beyond that point on the time horizon with appropriate placeholder user stories for future elaboration.

Philosophically, the big difference lies with a traditional focus on what will be done versus an agile focus on what to do when. The traditional planning framework is task-centric and the agile framework is value-centric. Having the Product Owner prioritize the user stories is central to the process. Doing so creates stakeholder engagement and enables value-driven delivery cycles.

Process Overview

The Agile Process Map™ introduced in Chapter 2 (Figure 2.16) illustrated the overall process flow of Agile Project Management. That flow diagram helped you create a mental map of the entire process of managing projects in an agile way. In this chapter, we will expand upon that flow diagram and drill down into the process of initiating a single, distinct project.

Initiating single, discrete projects in an agile fashion is a micro version of the macro view that you learned in Chapter 2. The process requires taking input from stakeholders and setting priorities according to their values. In order to do that successfully, the agile framework recognizes and responds to the demand to engage stakeholders in structured, meaningful discussions helping them clarify and articulate their values and priorities. The output of collecting, refining, and prioritizing input from stakeholders, customers, end users, and the team is a product backlog, as you can see in Figure 3.1.

![Figure 3.1 The Product Backlog Consists of the Features that are Prioritized and Groomed by the Product Owner.](image-url)
Although primary responsibility for creating the product backlog belongs to the Product Owner, it is done in collaboration with the agile team. The product backlog represents the vision for the entire product as decided by the customer/proxy. It is continually being improved as new insights are gained and documented, typically in the form of user stories. The agile values of transparency and adaptability underpin this approach.

In Scrum, as well as most other agile frameworks, the process of negotiating the iteration backlog is done in an iteration planning meeting. A portion of the product backlog is selected for the iteration backlog through a negotiated process where the agile team and the Product Owner discuss and refine their mutual understanding of exactly what will be delivered at the end of the iteration. The negotiating process during the iteration planning meeting usually has two steps.

In Step 1, the Product Owner describes what she/he feels can and should be included in the iteration. It is helpful to note that prior to Step 1, as part of the Agile Project Management process that occurs parallel to, but outside of, the iteration specific practices, the Product Owner spends time with the team to size user stories that are high on the prioritized product backlog and on the near-term time horizon. Using that preliminary information regarding size, the Product Owner presents what they believe can and should be included in the iteration. During Step 1, after the Product Owner describes what should be included in the iteration, the team clarifies the exact meaning of each user story. In particular, they want a definition of done for each story that includes specific metrics or acceptance criteria. Based on the mutual understanding that has been established, the team makes a soft commitment to a specific set of features for the iteration backlog.

Step 2 follows directly after the soft commitment while the Product Owner is still immediately available to clarify any unexpected, additional questions. The team discusses the approach they will use to create the deliverables, decomposes the user stories into tasks, and performs detailed estimating of each task. In mature agile environments, the team also performs a capacity analysis to confirm that enough hours are available from each of the required resources to be able to successfully achieve the iteration goal. Once the team has finalized their analysis and agreed they can succeed, they make a hard commitment to the specific set of features that will be delivered to the Product Owner at the end of the iteration. The hard commitment signals that the iteration has been initiated.

The process of collecting and refining user requirements into the product backlog then negotiating an agreement about which requirements are the highest priority and will be delivered at the end of the iteration, is the process that creates an environment where stakeholder engagement is possible.

**Identify Stakeholders and Build Engagement**

Every project comes into existence because of the action of one or more key stakeholders. Many project managers get into trouble because they assume those initiators are the only stakeholders with which they need to be concerned and they couldn’t be more wrong.

It is important to identify all of the stakeholders because they hold the power to provide or deny access to the resources needed by the agile team to fulfill the vision of the project. Many times key indirect stakeholders are overlooked. In doing so, many projects have run into major, potentially fatal, obstacles or even sabotage from stakeholders whose participation was required, but missed because it was not recognized.
In order to accurately identify all of the stakeholders for a project, a clear vision of the product, regardless of whether it is tangible or intangible, must be defined. Three of the tools commonly used in the process of defining and clarifying the vision for a releasable product are the *product vision box*, the *project data sheet*, and the *flexibility matrix*. The product vision box and the project data sheet will be explained in depth in Chapter 7, Iterating projects – Part 2, but we will define them briefly here.

A **product vision box** can be thought of as the product box on a store shelf. It has whatever graphic images and narrative content is necessary to convey to the customer what they can expect from the product. Significantly, it is in end user language and not techno-jargon.

A **project data sheet (PDS)**, by comparison, captures a project’s objectives in a minimalist document. Typically it is a one-page summary of the key objectives, capabilities, and information needed to understand the purpose and progress of the project.

A **flexibility matrix** is a simple tool that helps the customer/proxy clarify how to handle the unavoidable tradeoffs that may arise in the future and communicate that perspective to the team. The matrix clarifies which constraints are flexible and which are not, hence the name. It allows the customer/proxy to establish a top-level decision tool for making tradeoff decisions when the typical resource, time, or cost conflicts arise during execution.

As the customer gathers information about the project vision, she solicits customer views on tradeoffs and documents the results in the matrix. Because the matrix will be used throughout the project to decide tradeoffs, getting it settled early in the project is key to avoiding difficult or impossible negotiations later. Once people decide their position on a specific tradeoff, the effort to get cooperation becomes much more difficult without a flexibility matrix.

As stated earlier, philosophically the agile framework is process-centric. The **process goal is to create stakeholder engagement and value-driven delivery cycles**. If we want to build agile products, we need adaptive teams that include all stakeholders. Stakeholders are referred to as the customer and represented by the customer/proxy.

### Identify Stakeholders

Many consultants describe listing stakeholders and their interests in a project as the “most obvious secret to success” that is commonly overlooked. They know a great many budget-busting, project-imperiling change requests occur simply because the team lacked a written list of stakeholders and their interests. In fact, whole categories of mistakes can be avoided if this simple, yet profound, step is taken during development and before system delivery.

The stakeholder list includes those directly involved, like the client, the end users, management, channel partners, and supply-chain vendors. But it also includes those indirectly impacted, like technical or logistical support and interest groups.

The identification process can be rather informal, but it must be serious and reasonably thorough. It must include the favorable stakeholders, like project champions, the opposing stakeholders, sometimes called parasitic participants, and the independent stakeholders who can be influenced to support the project by the actions of the team.
In order to be valuable, the identification process needs to answer questions that will spur the team to think broadly about all of the stakeholders. Questions like:

- Who will the solution benefit?
- Who may be burdened by the solution?
- What groups or organization will use the system?
- How will the solution, and future improvements, be delivered?

Once the stakeholders have been properly identified, the process of engaging them can begin.

**Build Stakeholder Engagement**

The process of building stakeholder engagement in an agile framework is based on assumptions that differ rather starkly with the assumptions in the traditional framework. The traditional approach, as illustrated in Figure 3.2, assumes that the path from the problem to the solution is a direct, almost stair-step, process. The evidence of many projects, however, suggests that most customers need an opportunity to interact with the solution in the early stages in order to clarify their own understanding of the problem and accurately define the best solution. Agile frameworks, by comparison, assume that customers must traverse through an ambiguous process where they move between experiences that sub-optimally solve their problem in order to find the experiences that optimally solve it.

![Figure 3.2 The Traditional Development Approach.](image)

While customers interact with the team, their focus oscillates back and forth as they find the real boundaries of their problem and clarify the solution to it. Along the way, the solution moves from what they thought at the beginning to what they really need. Stakeholders, because of the limits of their humanity, need to be meaningfully engaged with the deliverables in order to move through this *cone of uncertainty* and find the optimal solution. (See Figure 3.3.)
It is no wonder that many traditional projects have had the sad experience of delivering precisely what was specified only to have the customer say it is not what they wanted. Sometimes it is the IKIWISI syndrome, sometimes it is the changing competitive landscape, sometimes it is the changing internal political landscape, but most often it is a combination of all three factors, causing the best solution to be imperceptible at the beginning of the project.

Two corollaries of the agile approach to finding solutions are point-based engineering (PBE) and Toyota’s set-based concurrent engineering (SBCE). PBE is used to encourage experimentation in product development and approaches product design as a sequence of decisions, narrowing and focusing prior decisions for subsequent ones, thus refining a broad marketing vision into a releasable product. SBCE maintains groups, sets or collections of design options throughout the design process and defers design decisions to the latest, responsible moment along the way to a releasable product.

All three approaches – agile, PBE, and SBCE – recognize that customers and product development must both work through a cone of uncertainty in order to identify the best possible solution.

It is impossible to be an Agile Project Manager without acquiring and maintaining a clear perspective of the key stakeholder values. Interestingly, the idea of key stakeholder values includes both the values of key stakeholders and the key values of all stakeholders. It is important to solicit, remember, and understand both and it is impossible to do so without a framework that engages stakeholders in participatory decision making.

This does not, however, mean creating reams of documentation. What it does mean, typically, is a change of perspective. Consider this example. If you were taking a cross-country flight, would you prefer the pilot (a) to have reams of information about the weather and other risks and not update via radio or (b) to have adequate information – but less than in “a” – and update it continuously as the flight progresses? Airline companies have proven that choice “b” leads to much better and safer results.
The agile approach to acquiring and maintaining a clear perspective of the key stakeholder values is referred to as “good enough” or “barely sufficient”. Neither of those terms imply that it can be insufficient. Insufficient is unacceptable. Overdone – think of it as gold plated – is waste. Bare sufficient is just right! Documentation is important to every development project but it is not a risk reduction strategy. Active involvement with customers plus adequate, bare sufficient documentation is the best way to achieve the project goals.

Additionally, when a Test-Driven Development (TDD) approach is applied then the test documentation supports two valuable goals. First, giving customers and the team clarity about architectural and design requirements they specify, and second, validating the deliverables.

This agile strategy of deferring the creation of documentation until the last responsible moment means everything from system overviews to user stories, from end-user manuals to marketing information aligns with what was actually built. However, this doesn't imply that no documentation is done earlier. Notes, rough drafts, sketches and other sorts of documents are created throughout development so that critical information isn’t lost. Those notes simply get refined for final delivery at the appropriate time.

Waiting to document information until it has stabilized eliminates both cost and risk. Avoiding any unneeded rework reduces cost. Risk is reduced because the documentation is current and accurate.

Another advantage of this approach is that it engenders stakeholder trust because they are engaged by an experience of having solid information successfully built and transferred.

This is so because the active involvement with customers required to accomplish it implies that face-to-face (F2F) communication, even if it is via Skype, is happening.

Again, notice that the barely sufficient approach described above means doing what is necessary to achieve success, but no more. So if the customer cannot attend in person then the next best mode of communication, in this example Skype, is accepted as sufficient.

Interestingly, because this approach helps the team deliver a quality product at regular intervals it acts as an intrinsic motivating reward. In the same way that seeing the product vision realized motivates customer participation, stakeholder engagement inspires team commitment to the project.

Acquiring and maintaining a clear perspective of stakeholder values requires an Agile Project Manager to develop active listening skills and apply them professionally.

We have all heard that a good listener strives to completely understand what another is saying. In fact, an Agile Project Manager must not only understand what is being said, but more importantly, what is meant by what is being said. In the end, an Agile Project Manager may disagree with whether the stakeholder’s need is real, but before they act to manage the situation, they must know exactly what is requested. Therefore, listening is one of the most useful skills agile PMs can have. How well they listen has a major impact on the quality of stakeholder engagement.

Active listening intentionally focuses on who is speaking in order to understand what is said and what is meant. As the listener, the Agile Project Manager is then able to document a user story to the stakeholder’s satisfaction. Doing so improves mutual understanding and leads to the transparency and adaptability required in Agile Project Management.
In order to improve your active listening skills apply the following model:

**STEP 1 – Receive.** Remove any barriers - physical or mental - that may limit the attention you give to hearing the message. This may mean asking the stakeholder to wait a moment while you complete a task so you can give them your undivided attention. It may mean moving into a quieter or more private location so important or confidential information can be shared.

**STEP 2 – Analyze.** Deciphering, sometimes called unpacking, the stakeholder’s words involves analyzing the speaker’s vocabulary and word usage to correctly interpret them while decoding their non-verbal signals. It also means taking care to confirm context variables – like cultural norms, technical baselines, and legal requirements – in order to prevent any misunderstanding.

**STEP 3 – Evaluate.** Only after being certain to receive and analyze the stakeholder’s message, should it be evaluated for missing information, which is needed in order to have a clear and complete picture. It is also at this step that the Agile Project Manager decides what options exist to appropriately handle the information received.

**STEP 4 – Handle.** Only after ascertaining that the message was accurately received and appropriate options were analyzed, is the final step taken – handling the request. Handling may be a verbal response to a stakeholder’s concern about an issue or a physical response, such as documenting a story card for a future more in-depth discussion. Both let the speaker know you have gotten their message and how it will be addressed.

Note that the first letter of each step – R, A, E, H – spells “Hear” in reverse.

**Value-Driven Delivery**

**Document the Business Case**

A business case is a written document that explains how the use of resources is aligned with the accomplishment of a goal or the implementation of a needed change. A compelling business case leaves no doubt that benefits far outweigh the costs and risks. (Teaching exactly how to create one is outside of the scope of this book, but a summary explanation is appropriate and included here.)

The acceptance criteria for a business case include:

1. Providing the context and background of the business environment and need
2. Outlining the options for meeting those needs, and recommending a solution
3. Defining success metrics for the proposed solution and an analysis of the related cost\benefit and financial ratios
4. Presenting a compelling case for change that secures executive support
5. Securing executive approval prior to moving forward

The business case should include a business model snapshot showing:

1. Critical success factors and goals
2. Process maps of the current and future processes, including:
   - Roles and responsibilities
• Business rules
• Workflows and/or Swim-lane diagrams

3. Opportunities for process improvement

The process map is a high-level visual picture of the major process steps down to the individual sub-processes. However, individual sub-process steps are detailed within a procedures document. It is used to improve understanding of business processes by breaking them down into simple visual steps in order to better identify potential problem areas, gaps and opportunities for improvement. An effective process map is not only accurate; it must also be easy to read and understand.

The goal is to define the problem the project seeks to address and identify impacted stakeholders, systems and business processes. In so doing, the scope of the project is clarified and indirect stakeholders are often identified.

**Write Contracts**

As a solution provider or customer or supplier, at the beginning of a project there is too much at stake to work without a contract. No matter how much the Agile Manifesto values customer collaboration above contracts, legal departments the world over are going to insist contracts are necessary. In fact a contract is just a way to define the rules for working together and communicating. Good rules increase the probability of success for both parties, while bad rules inhibit successful progress. So the question is not, “Contract or no contract?” The question is, “Which contract best conforms to the needs of an agile project?

To answer that question, we must consider both the purpose of the project and the context of its delivery in order to define the contents – the rules – that need to be in the contract. Because one size does not fit all, the best approach is to understand various criteria for evaluating agile contracts.

One basic agile principle to keep in mind is that contracts produce no added value for the customer. They are simply a necessary formality which can be used to create a win-win relationship with customers through good communication. But, by definition, contracts are a waste product, so any time spent negotiating and writing them should be optimized to improve communication and reduce cost.

Also keep in mind that a contract divides shared risk and defines trust points between the parties. Who pays how much if things are more difficult than expected and who benefits how much if the project goes well both hinge on those trust points and must be articulated well. Unrealistic promises or demands, poorly defined time frames or functional expectations, and win-lose tactics all undermine the probability of project success. The most common result of an inequitable contract is compromised quality, which leads to suffering for both parties.

**Criteria for evaluating contract suitability include:**

1. Does the contract include defined project objectives?
   - Is the elevator statement and product vision included (or promised as an addendum with a time constraint)?
   - Is the project data sheet format included (and promised when complete as an addendum with a time constraint)?

2. Are key roles and responsibilities defined to foster cooperation?
   - Is the agile framework and/or process defined?
• Is the Product Owner designated, with a clear statement of responsibilities?
• Are the key team roles – and any specific personnel – defined, with a clear statement of responsibilities?

3. How is scope delivery defined, and how is it linked to invoicing and payment?
• How will scope be managed? (i.e., Backlog grooming and by whom; Iteration-level addendums)
• Is the contract “staged” with checkpoints for visibility and customer go/no-go decisions?
• Does the contract include an “end of iteration, cancel-for-convenience” clause for both parties?

4. How are risks and rewards divided between customer and provider?
• Is cost responsibility shared using a target cost principle (like at Toyota) so that when the target cost has been met, both parties bear the excess cost of changes?
• Are bonus, penalty, early termination and late delivery clauses documented?

Examples of possible contracts include:
• Agile Iteration Contract
• Time and Materials Contract
• Phased Development

**Agile Iteration Contract**

**Scope Management:** Mutual, based on the team delivering agreed upon features to defined quality standards by iteration end, and the Product Owner not changing the iteration backlog before iteration end.

**Risk Management:** Mutual, using product backlog grooming and iteration backlog commitment negotiations. Scope can only be varied between iterations and within the boundaries of the overall project scope.

**Communication Management:** Project scope is confirmed at the start of each iteration in order to build trust. The iteration scope is documented as an addendum to the contract also.

**Invoicing & Payment Management:** Either based on a time and materials agreement or an earned value agreement, usually with a not-to-exceed cost ceiling for the iteration or release.

**Time and Materials**

**Scope Management:** Not defined; dependent. When the customer doesn't see value they can decide to stop paying so the project comes to an end. A cooperative effort, driven by the customer's limited budget and the provider’s desire for more business, focuses both parties on creating value.

**Risk Management:** Customer carries all change management risk. The budget may be used up without achieving expected business value and the customer may not get everything that was sought. Insuring that only valid effort and expenses are invoiced and paid is important.

**Communication Management:** Interdependent need to actively prevent dissatisfaction.

**Invoicing & Payment Management:** Work for an agreed upon time period then invoice the customer. Variations may include a cost ceiling to limit the customer's financial risk.
Phased Development

Scope Management: Scope is typically funded on a quarterly basis and additional funding is approved following each successful release. Knowing another release will occur, next quarter, or not, guides feature selection for the timebox.

Risk Management: Mutual, but limited to one quarter’s development costs and a second quarter’s budget (contingent funding).

Communication Management: Interdependent need to secure additional funding approval drives both customer and provider to insure each release is successful.

Invoicing & Payment Management: Invoicing for agreed upon work is paid within funding limit using iteration contract addendums.

The contract is a very important foundation for a successful project. The Agile Manifesto was spot on when it stated, “Customer collaboration over contract negotiation.” Contract negotiation is fundamentally important and must be done. But it should be done with the perspective that a vibrant working relationship between the parties must be a result that is more important than the contract document.

Adaptive Planning

Acquire the Right Team

Earlier we said that Agile Project Management is not a silver bullet because without the right skills on the team, nothing can help. Later we noted that effective risk management hinges the actions of cross-functional teams and that the agile ethos recognizes teams must be trusted in order to optimize the chance to identify risks and reduce errors. We also pointed out that the seventh principle in the LSD framework is “Optimize the Whole” by always using cross-functional teams so important or critical facets of the problem aren’t overlooked. So the question is not, “Can we have a cross-functional team with the right skills?” We know the answer is that we must have a cross-functional team with the right skills. The question is, “How do we go about getting that particular team?”

The first part of the answer to that question is actually a reality check. If the project is number 103 on the company’s priority list then the Agile Project Manager must frame the team requirements within that reality. The crucial key is to use the business case and flexibility matrix, described above, to solicit the executive sponsorship needed to clarify the tactical reality with the Product Owner.

The second part of getting the particular team the project needs requires evaluating potential (or assigned) team members according to key factors such as:

- **Ability** – What specific competencies do they provide?
- **Availability** – What is their availability and what are their competing commitments? Are they local or remote?
- **Cost** – How appropriate is their cost given the budget constraint?
- **Chemistry** – How well do their work style preferences align with the team culture and environment?
- **Experience** – What similar or related work have they done? What time and quality constraints were the prior project under?
In many, if not most organizations, whether they use a traditional or agile framework, one of the biggest challenges to getting the proper team is pre-assignment. Pre-assignment of team members is necessary when the project is dependent on specific expertise and that expertise is in short supply. Sometimes pre-assignment is defined as a part of the contract and must be honored. But it happens many other times when there is limited rationale for accepting it. In those circumstances – and maybe in the situations just described – the project manager must evaluate carefully whether critical skills are missing from the team and take steps to correct the problem.

It may also be necessary for the project manager to act in order to insure the project receives competent staff on a timely basis, and team members don’t have their bandwidth constricted due to new work assignments. A professional Agile Project Manager must be willing and able to negotiate as toughly as required to provide for project success.

The desired result is the right person in the right role at the right time. Until the team has that result, the Agile Project Manager must continue to negotiate and secure those people for the project team.

**Project Kickoff Meeting**

In traditional project management, most projects start with a kickoff meeting. In an agile framework it is necessary to do likewise, however, the way the meeting is conducted and its goal are dissimilar from the traditional approach. An agile project kick-off meeting has several key objectives that provide the framework for an agenda, including:

- Establish ground rules
- Have the project vision presented by the customer/proxy
- Review the high-level business case
- Document the one or two dozen key features
- Project setup and release and iteration planning

An agile project kickoff meeting begins with identifying the agile framework that will be used and agreeing upon behavioral-collaboration rules the team expects of one another. Because each team and the context of each project is different, this step articulates what all team members can expect while working together on the project. Discussion of development standards and team member roles are included. Experience has shown that skipping this step will cause issues throughout the project, starting with the next parts of the kick-off meeting.

One rather interesting difference between an agile kick-off and a more typical traditional one is that the team helps the customer/proxy clarify their vision through discussions and important group exercises. The exercises include developing the product vision box and writing the **elevator statement** (which is covered in Chapter 7) and the project data sheet. The goal of the meeting goes beyond presenting the “big picture” and drives to getting every team member aligned with a clear understanding of what is being developed. The desired outcome is a clearly defined scope that is documented at a very high level in one or two dozen key features. Those are all very large pieces of information with minimal detail, but clear connections to one another and to the strategic objective embodied in the business case. The farther out on the time horizon the feature is, the broader the feature definition. The purpose is to orient the new project team to the business reasons driving the project.
Typically the kick-off meeting, to this point, has run about two hours and senior stakeholders start to get anxious for it to end. However, the team needs more time, up to a full day depending on the size of the project, to effectively finish the process. Therefore, a common practice is to publish an understanding that when this point is reached in the agenda, there will be a short refreshment break and senior managers may leave. Project set-up and release and iteration planning begins after the break and may take the rest of the day.

Planning project set-up involves defining environmental needs for getting the project off to a good start. Whereas release and iteration planning involves identifying any required proof-of-concept work as well as laying out subsequent iterations. Defining the probable total iterations allows the customer/proxy to make needed future planning decisions regarding feature priority.

When teams are not colocated, every effort should be made to bring them together for the kick-off meeting. Agile emphasizes face-to-face communication because of its many benefits. But given the reality of remote teams and budget constraints, this is the time for selective use of available resources. The kick-off meeting should receive high priority because the shared decisions will guide and focus the team throughout the rest of the project.

**Apply Incremental Delivery Cycles**

The “early and often” principle of iterative development has often been called “deliver fast,” but we feel it is better to think of it as “remove delays.” The focus is on adding value to the customer without delay and since delays represent waste, removing them will result in faster delivery. While the benefits of delivering fast are clear, it is essential that this is done in a sustainable manner.

**Using Adaptive Planning Norms**

*Adaptive planning,* as commonly practiced, entails incremental delivery cycles or iterations. Each iteration is a single timebox during which specific, agreed upon deliverables will be constructed and demonstrated to the stakeholders.

In order for this approach to work effectively and efficiently, it is imperative that norms and expectations be defined, communicated, and understood. Then they must be rationally and consistently applied. That means key variables must be consciously chosen and clearly articulated.

The first of those key variables is the length of time for each timebox. The balance point for choosing each length is providing desired flexibility while avoiding unneeded cost. Because each iteration begins and ends with specific planning and review activities, there is a type of “overhead” that is fixed and must be carried by the work delivered during the iteration. This means that if the work accomplished per team member, per day is essentially the same for both a two-week and a four-week iteration, then the planning and review costs – which will occur twice as often for the two-week iteration – will in fact be twice as high.

This does not mean that longer iterations are automatically more desirable. Iteration length should be chosen to optimize the amount, frequency, and timeliness of feedback shared by the agile team and the customer/proxy. Since the feedback loop is a key control, limiting the risk of the team spending time and effort incorrectly developing a product or service due to a misunderstanding of the customer/proxy vision, shorter iterations may save a significant amount of rework.
An additional aspect to consider when choosing the iteration length is how it may, or may not, facilitate customer engagement. Customer and key stakeholder engagement is critical to successful Agile Project Management and customer engagement entails a certain amount of needs gratification. So, depending on the organizational audience, shorter iterations may be advisable if that keeps them excited about and engaged in the progress being made to fulfill their vision. Remember, the value of that engagement and feedback must be weighed against whether it is less than or more than the “cost” associated with shorter iterations though.

Ultimately there are trade-offs in every situation, but a norm must be established because having consistent timeboxes enables the team to develop a rhythm that increases both speed and quality.

**Defining Timeboxes**

As we said earlier, there are several types of timeboxes employed in Agile Project Management. The roadmap is most equivalent to program level planning and is decomposed into release plans, which are most equivalent to project schedules in traditional project management. Release plans are composed of iteration plans. The iteration plan is a combination of a timebox and work effort (See Figure 2.6). By definition, the size of the timebox for an iteration is stable and typically defined as either two, three or four weeks.

How to prepare for the first iteration since it’s the first timebox is a common question. Everyone recognizes that beginning a project requires certain resources and without them we move towards project failure, not success. Prior to the first iteration, a clearly defined vision is the first thing needed to identify, at a high level, the types of work the team will be required to accomplish. The vision is simply decomposed into the categories of work, not the details, and the general degree of skill required in each category so that a proper cross-functional team can be defined.

The second thing needed is a team with the right knowledge, experience, skills, tools, and processes to effectively produce the deliverables. A colocated environment is preferred and it must have at least the minimum tools and workspace (even virtually) so the team can function.

The third requirements protocols for creating and supporting visibility amongst the team as it creates the deliverables. We will discuss communication requirements in more detail shortly.

And lastly, high-level architectural outlines are needed to facilitate planning for at least the first two iterations (aka “**Just Enough**”), handle questions when needed during those iterations (aka “**Just In Time**”), and meet any regulatory requirements (aka “**Just Because**”). The architectural outlines must be adequate to guide emergent design and incremental delivery of business value.

**Create Team Performance Environment**

**Recruit Coaches**

It might seem strange to start talking about team performance by suggesting a coach since many agile teams work without ever having a coach. However, research clearly shows un-coached teams are much less likely to achieve optimal performance. For example, it is generally expected that the average Scrum team will manage a 35% productivity improvement. Compare that with the various properly coached teams that experienced 300% to 400% improvements at Yahoo, according to Gabrielle Benefield in *Rolling Out Agile in a Large Enterprise*.8
An agile coach, whether they are an internal company resource or an outsourced expert, has three key roles. They act as a trainer, a consultant and an advisor who extends what was learned in classes to the specific environment the team will encounter. The coach is there to help the team apply an agile framework to the particular setting and impediments they face. Sometimes that means adapting the framework while other times that means challenging existing norms.

Top notch coaches aim for knowledge transfer to the team who is responsible for making the work happen. They focus on facilitating teams to create change and improve delivery. They understand they must help the team rethink basic assumptions and mental models about their environment and quite often focus on making individuals and teams solve their own problems using non-directive techniques.

Internal coaches have the advantage of knowing the corporate culture and history as well as the team. External coaches often bring a new perspective and broad-scope process customization expertise that enables them to challenge assumptions and suggest alternatives more comfortably. Some situations are best served by one or the other.

Understanding Agile Leadership

In the traditional environment, the project manager utilizes directive, command-and-control tactics, such as assigning tasks to team members. The Agile Project Manager, however, relies on facilitation efforts and servant leadership to help the team achieve its goals. Servant leadership includes a number of skills and techniques, all aimed at developing and facilitating the agile project team. They include:

- Creating an environment of personal safety where team members may both innovate and come to friendly disagreements; one where the Agile Project Manager and the team support the individual.
- Mentoring team members on the agile framework as well as on general management and technical skills, allowing the team to become cross-functional and self-organizing.
- Facilitating (and not controlling) team meetings, including release planning meetings, daily stand-ups, demonstrations, reflection workshops, reviews, and retrospectives.
- Guiding the team, when necessary, to foster appropriate value-based decisions.
- Removing obstacles that impede progress or facilitating the effort of team members to do so.

As the agile team matures, it moves along the continuum from the directed traditional team to the self-organizing agile team that cross-functional, and highly motivated to succeed. When the team reaches this state, the Agile Project Manager can act largely as a consultant, serving the team as a facilitator when called upon. Ideally, the Agile Project Manager will not conduct individual performance reviews as this is considered a conflict of interest, violating the servant leadership role.

The Agile Project Manager also fosters adaptive leadership within the organization. Such efforts include:

- Adapting to (and, in the agile spirit, embracing) change. On projects, many things are subject to change, including requirements, priorities, customers, workforce, and organizational goals. The Agile Project Manager and the team, in consultation with the customer/proxy, must decide which changes to respond to during an iteration, and which to defer until an iteration is complete. “Locking down” an iteration is an example of a decision to prevent unapproved scope changes that ultimately escalate costs.
Adapting actions to guide agile process outcomes. A team may implement improvements based on interim reflection workshops and retrospective meetings. Such adaptive actions may include things like reducing features, adding an iteration, creating another agile team, or identifying a new metric.

Adapting the agile framework to the work environment and the customer/proxy. For example, over time, an agile team may decide to customize its numbering schema for estimating story points instead of adhering strictly to the Fibonacci sequence.11

Part of adaptive leadership can best be described using a term that has become both a cliché and a source of contention - emotional intelligence. But it is important for an Agile Project Manager to not be distracted by that word choice because, as Jim Highsmith points out, “Management research shows that mood or “emotional intelligence” in leaders has a much larger impact on performance than we may have imagined.”12

Popularized by Daniel Goleman, emotional intelligence (EI) distinguishes an effective leader from a merely competent one and is characterized by the presence of its components - self-awareness, self-regulation, motivation, empathy, and social skills.13

The Agile Project Manager must recognize that their emotional intelligence is contagious and will dramatically impact the team’s success. Because the team will experience both highs and lows, quite often with unexpected volatility, during the project, supporting appropriate responses and discouraging inappropriate ones is critical. Group dynamics that are conducive to creating desirable, emergent results at the edge of chaos where most teams must work are fundamental to optimal productivity.

A best practice technique used by many professional Agile Project Managers is defining the team’s rules of engagement, or ground rules. The rules establish norms and expectations for team member’s treatment of one another. The team discusses, develops, defines, documents, and posts the rules in a prominent place. Then it enforces and adapts the rules over time. Doing so fosters an interdependent accountability based on self-discipline.

The rules do not discourage the healthy, necessary conflict and contention of emergent design, but foster and direct it in positive ways. Great teams feed on the energy of diverse ideas in contention to produce the highest quality results.

Examples of common rules of engagement include:

- Everyone participates
- Respect differences
- Attack issues, not people
- Everyone has an equal voice
- Everyone has valuable contribution to make
- Honor confidentiality and privacy within the team

Cultivating an agile team spirit enables exciting productivity. It does so in part because the healthy contention of emergent design brings important risks into focus so the team can mitigate and resolve them.
Clarify Risk Considerations

Organizational Practices
During project initiation, the focus of risk management is to enable the team to make realistic, long-term commitments. Despite the uncertainties that surround every project, customers need reliable schedule commitments so they can integrate the deliverables into their business plans.

Risk management must take into account both the generic risks that are common to all projects and the unique risks specific to the project. A best practice first step for doing so is to have the team – preferably including the customer/proxy – brainstorm a list of possible catastrophic events that could impact the project. While it is not likely that any of the catastrophes will happen, they hold the seeds or root causes that will manifest themselves in smaller risk events.

Creating the list should be an exercise that is simultaneously serious and fun. A large dose of good natured conjecture will produce a broad range of ideas, which can be sorted out afterwards. After gathering the team and passing out index cards, encourage them to use dynamic, dramatic, negative futuristic thinking to write down and call out any notion that might lead to project failure.

Writing any of the following common questions on a flipchart or whiteboard where everyone can see them will help stimulate the team’s thinking.

• When you have a nightmare about the project what is happening? *(Academy Award for Best Screen Play goes to the winner.)*
• Imagine that a year after the project’s disastrous demise you are on *60 Minutes* being asked what went wrong. Describe what happened? *(Pulitzer Prize in Investigative Reporting goes to the winner.)*
• Visualize the best possible outcome for the project. Can you describe the opposite? *(Screen credit as a Co-Creator of an episode of The Simpsons goes to the winner.)*
• Can you describe a scenario of utter project failure and avoid blaming anyone? *(Teflon Don award goes to the winner.)*

The point is to inspire serious fun while digging up possible risks by reading index cards aloud encouraging further insights and team collaboration. Reading the cards aloud can be assigned to a neutral facilitator if the team prefers anonymity.

Once the list of catastrophic scenarios has been brainstormed, the follow up is to analyze them for possible root causes either immediately after or in a subsequent meeting.

The risk analysis can typically be done by a smaller cross-section of the team, freeing up the remainder to work on other tasks.

The final step in the initiating processes to create a risk matrix that shows the probability of a risk occurring and its impact, usually at the high, medium, and low level of granularity. Some teams extend this first assessment with numerical values that are calculated by multiplying the risk probabilities by the rough order of magnitude of risk the impact. During the planning process, this basic risk profile will be expanded and detailed, so it will be covered in more depth in Chapter 4.

The focus of these organizational practices is to help teams focus on avoiding risk by identifying them early. One particular category of risk is regulatory and it needs a fuller discussion.
Regulatory Discovery

Many organizations must comply with governmental regulations. Compliance is mandatory so implementing an agile framework must include documentation that auditors judge to be at the “right” level. Mike Cohn made a good point of this when he developed the idea of a “Relative Penalty” in weighting priorities. He shared how one of his clients “joked that the only downside to not implementing the feature was that the CEO might spend some time in prison.” Because she wasn’t the CEO, the feature had a relative benefit of 1, but a relative penalty (i.e., risk) of 9 for the CEO.

So part of understanding customer needs includes identifying all stakeholders, internal and external, like audit staff and regulators, and insuring that the team is made aware of audit needs, like documentation versions and trails.

Documenting user stories can often still be done on index or note cards provided there is a proper protocol for tracking them throughout their lifecycle for auditing purposes. In addition to tracking the stories, there may be a requirement to track testing plans, results and constraints along the continuum of rights, with a trail such as read, read/write, and read/write/change for regulators like the Food and Drug Administration (FDA).

An Agile Project Manager must guard against the common organizational drift towards compliance as an implicit, primary focus. Compliance activities mitigate risks like financial fraud, product liability, and construction defects. Compliance is legitimate work, but project managers are responsible for exercising oversight and challenging customers when they lose sight of real value driven by real need and slip into wasteful drift. The best strategy is to minimize wasteful compliance and then direct needed compliance away from the critical path, critical team members, and critical resources.

Compliance isn’t a zero sum game as long as the Agile Project Manager does reasonable discovery and exercises discretion to produce only documentation that contributes to deliverables, now and in the future, and fulfills required compliance mandates.

Lynne Nix, founder and president of Knowledge Structures, has a saying summing this up perfectly. She says, “A little documentation goes a long way… if it’s the right documentation.”

Defining Quality Standards

Quality is inherent when an agile framework is implemented correctly because only working solutions are delivered. It is also the first “shortcut” teams turn to when making progress becomes daunting. Therefore, it is crucial during project initiation to define the quality standards that must be applied. That means appropriate definitions of acceptance test requirements for each class or category of artifacts is needed so the team will know what level of validation their work will require.

A common challenge for quality professionals is adjusting work processes to fit the collaborative, iterative, incremental manner in which deliverables are created. Specifically, because agile frameworks focus on upfront quality-in design and development, the level-of-effort will be higher than customary. Balancing that early investment are the later savings when there is less need for quality assurance activities, such as reviews and inspections. Also, because deliverables emerge, quality professionals must adjust to what is sometimes perceived as “incomplete” artifacts in the early iterations and help find ways to work with in-process artifacts.

Another challenge is defining standards for the iteration review meetings. The review meeting shows the working solution at that time and demonstrates explicit progress toward key stakeholder
goals, so standards are usually pretty clear. What is often less clear are expectations regarding stakeholders providing feedback the team can use. If the demo occurs in a regulated environment, a more formally defined process may be needed to satisfy quality criteria.

It is well known that the cost to fix a defect rises exponentially the later it is discovered. Following standards and guidelines is an important quality technique that leads to greater consistency and work quality while reducing cost by eliminating rework. Agile frameworks recognize this and promote doing so as a best practice. Agile teams should be following enterprise-level guidelines, which should be tailored versions of industry guidelines, in order to reduce the likelihood of poorly conceived quality standards.

**Provide Communication Support**

**Colocated and Distributed Teams**

Whether an agile team is colocated or distributed, communication support must be provided by design and not by accident. Colocated teams enjoy a host of advantages and require only normal levels of support. The real challenge is in supporting communication for distributed teams. Key factors that must be considered include:

- Synchronizing communication
- Enabling collaboration
- Providing enough communication bandwidth

Remember that the foremost purpose of the daily meeting is help the team synchronize their communication. The most common challenge for remote teams are the verbal language and accent issues due to the lack of visual clues. Because so much communication is nonverbal, the lack of visual clues creates a huge risk of misunderstanding. Two solutions to this challenge are, first, to invest in a good headset (and/or hearing aid!) because vocal clues are very subtle, and second, use detailed written communication preceding the call. Because English is the lingua franca of many business teams, and because not everyone has a good “ear” for hearing past the accents of those speaking English as a second language, encourage the use of detailed written communication that can be distributed before the conference call or tele-meeting. Doing so allows other teammates to use the written word to develop an “ear” for the spoken word, vastly increasing the quality of communication occurring.

Remember that we live on a round globe with an ever moving cycle of night and day that works against enabling collaboration because of time zones. There are many ways to deal with this issue including shifting core hours to find an overlap and using asynchronous communication like emails and wikis. But a proven best practice is to modularize using a development hierarchy that distributes project-level pieces first, then theme-level pieces second, team-level pieces third, followed by feature-level pieces, and lastly function-level pieces (i.e., Developers, QA, Testers, etc.).

Providing enough communication bandwidth to achieve actual knowledge sharing is also a challenge. Solutions to this challenge include rotating developers across projects, features, and modules, using groupware tools like blogs and wikis, and investing in commercial Agile Project Management tools.

**Participatory Decision Models**

Agile leaders seeking to optimize team performance also need to use a participatory decision-making model in order to make sure each team member’s voice is heard.
Participatory decision making is a creative process where ownership of decisions belongs to the team, and where finding effective options that everyone can support is the focus.

Over the last ten years, Agile Project Management frameworks have enjoyed a meteoric rise in interest because of the way they have improved operating outcomes for so many organizations. One of the variables easily identified as a significant contributor to that success is participatory decision making. As stated previously, the agile ethos recognizes that the solutions to the significant, complex problems faced by most organizations cannot be solved by human beings acting like biological machines. Organizations need the creative, non-linear, and imaginative insights that only come from fully engaged people.

Consider the situation as a puzzle in the form of a question. Who is best qualified and most responsible for the solution that will best meet the customer's needs? Is it best to have management interpret the customer's needs as an intermediary for the team doing the work or for the team to communicate directly with the customer?

For many decades, going back to Henry Ford, the answer from business has clearly been “management.” To Ford, management was much more intelligent than workers and only they could be trusted to decide how to improve the manufacture of cars. He had very little respect for workers and no concept whatsoever for the knowledge workers that currently deliver results in today's organizations.

Today's projects have a threefold difficulty with Ford's thinking model. First, while Ford's policy allowed for setting up a very good static process for building one kind of car, it offered no flexibility, an absolute requirement for achieving success on today's projects. Recall Ford's famous statement that people could have “any color they wanted as long as it was black.” That simply won't do!

Second, in Ford's workplace, management may have known better what was happening on the manufacturing line, but on projects, the knowledge workers dealing with the environmental challenges always understand how to solve the problem better than management, who is usually one step removed from the actual point of production. In Agile Project Management, the process is dynamic, always changing to meet an evolving challenge, as the team moves through the cone of uncertainty.

Third, and finally, when Ford was running his plants, a job's most important value was the paycheck that provided for the worker's family. Today, research has shown that monetary compensation is not the number one factor for most knowledge workers. The question of compensation has to be made neutral, but knowledge workers are also looking for engagement, challenge, and meaning in their work. Therefore, due to the massive investment in education and training required to make knowledge workers productive, any company using Ford's approach would be plagued by difficulties trying to keep workers motivated.

So if Ford's approach is no longer viable, what is the alternative? Peter Drucker emphasized the need for organizations to empower the knowledge worker to make decisions that they - more so than management - are in the best position to make; and also to avoid having the knowledge workers leave, damaging the organization. In Jim Collins's famous book, Good to Great, an outline of the answer is also identified. The great organizations he describes engage in rigorous debate, often over extended periods of time, using dialogue, not coercion, to interrogate the truth down to the brutal facts, in a way that allows individuals to be extremely interactive, until the best solution is identified. We also see it implied in Roger Martin’s article, The Opposable Mind: How Successful
Leaders Win Through Integrative Thinking, where he describes how real leaders, great leaders, tolerate and embrace ambiguity as long as necessary during discussion because they refuse to be limited by “either/or” choices.\textsuperscript{18}

The agile methodology and its many frameworks have embraced participatory decision making as the alternative to Ford’s thinking model.

Because the quality of the solution delivered by an agile team is based on the trust and respect needed to enable a free flow of information, a vigorous discussion and active participation by every member is needed for the participatory decision-making process to be engaged and fostered to maturity in the agile ethos. Experience has shown that if any of those key components is left out, ineffective, poor quality results follow. Experience has also shown that it is difficult to develop the sophisticated leadership skill needed to facilitate, influence, and coach a team into the healthy, durable relationships required for participatory decision making.

Participatory decision making either helps the team operate smoothly or mires it in a swamp of indecision, and the difference is largely dependent on who is leading. If the leader shies away from the discussion necessary to get the structural engineer to challenge the architect, then the building can’t be built cost-effectively. If the construction manager has to attend too many meetings, in the name of “coordination,” the project gets hopelessly behind schedule because the team doesn’t have access to the customer/proxy when needed. The ends of the continuum – too little and too much – can both cripple participatory decision making.

Leadership is critical to effective decision making in an agile project environment where thousands of decisions must be made using information that is often vague. Customer desires are unclear. Technology is untried in the exact situation. Eight or nine out of ten decisions can paralyze the team because the fuzziness makes them oscillate between choices. Often times, once the required, healthy, vigorous debate has occurred and the team has reached an impasse because the ambiguity engulfs them, the leader has to step forward. An effective leader acknowledges the ambiguity, takes responsibility for the impact of the decision – whatever it may be – and enables the team to resume productive activities by making the decision.

Fortunately, as Agile Project Management has matured, it has developed a number of tools and techniques aiding the team in achieving remarkably participatory decision making.

Earlier in this chapter, we said that most agile frameworks use a negotiating process to choose the iteration backlog during a two-step iteration planning meeting where the agile team and the Product Owner discuss and refine their mutual understanding of exactly what will be delivered at the end of the iteration. That negotiating process is a participatory decision making tool where the Product Owner describes user stories that should be included in the iteration, the team asks questions to clarify the exact meaning of each user story, and specific metrics or acceptance criteria are defined. That sterile description probably does not provide an accurate portrait of the vigor with which the actual debate occurs while a mutual understanding is being elaborated. Because the team will be making commitments based on the iteration backlog, they have a keen, vested interested in participating in how it is defined.

Earlier we also mentioned that prior to iteration planning meetings, part of the Agile Project Management process involved the Product Owner spending time with the team to elaborate and size the user stories that are high on the prioritized product backlog. Elaborating the user stories, as well as sizing them, are both also participatory decision making tools. We will detail the exact
process of writing and elaborating user stories when we cover Planning Projects. We will also detail the exact process of sizing them in the chapter on Estimating. For this chapter’s purpose, we will summarize the processes here in order to clarify how they act as participatory decision making tools.

By definition, all projects begin with incomplete requirements. The goal of sizing is to give the team and the customer/proxy a quick, relative measure of the effort involved with delivering a particular user story. The process, at a high level, simply assesses how big, complex, and risky a story is compared to other stories and then assigns it a value referred to as **story points**. Sizing allows for adequate and appropriate planning, focusing the team on what is needed to accurately estimate high priority components on a time horizon that is near enough to be accurately understood and applied.

The process of sizing integrates customer and team ideas about stories, subjects them to energetic discussion, analyzes them according to the customer’s vision, and begins to align them with time and other constraints. The result is decision-making that is unlocked by dedicated participation. This balances agility in decision making with ownership and accountability, a hallmark of participatory decision making.

**Invest in Continuous Improvement**

**Define Required Ceremonies**

During project initiation the seeds of continuous improvement are planted by defining required ceremonies. In Scrum there are three key ceremonies: Sprint Planning, Sprint Review/Retrospective, and the Daily Meeting. Other agile frameworks have variations that typically include these three and may add a few more. The important thing is to know which ceremonies are needed by the team and stakeholders and then articulate them during project initiation.

It’s important to remember that **agile frameworks intend to deliver the right product for the current need, which may be different than the need that existed when the project began**. That means agile teams must be open to, and embrace, the changes that emerge as they move forward. Ceremonies act as the gravitational force that keeps the team working to deliver the right product for the customer/proxy who defines the solution’s orbit.

The process of initiating projects in an agile framework is a prioritized, cyclical, iterative process that elaborates in detail only those things on a practical time horizon. Having the customer/proxy prioritize the user stories is central to the process and creates stakeholder engagement and value-driven delivery cycles. In order to do that successfully, the agile framework recognizes and responds to the need to engage stakeholders in structured, meaningful discussions, helping them clarify and articulate their values and priorities. The output of collecting, refining, and prioritizing input from stakeholders, customers, end users, and the team produces an actionable, high-value product backlog.
Chapter End Notes

2Ibid.
8The Fibonacci sequence was introduced Western European mathematics by Leonardo of Pisa, also known as Fibonacci, in his book Liber Abaci in 1202, although the sequence had been described earlier in Indian mathematics. It is a non-linear sequence where each subsequent number is the sum of the previous two. Over the centuries it has been used in biological sciences to describe branching in trees, fruit spouts on pineapples, and uncurling of ferns. More recently its applications have included computer search algorithms, graphs of interconnecting parallel and distributed systems. In Agile Project Management it is used in the process of defining the size of user stories.
12Knowledge Structures, Inc., 2031 Rockwood Drive, Sacramento, CA 95864