ADAPTIVE LEADERSHIP

ACCELERATING ENTERPRISE AGILITY

JIM HIGHSMITH

Forewords by GARY GRUVER, CHRISTOPHER MURPHY, *and* JOHN CROSBY

Adaptive Leadership

Accelerating Enterprise Agility

Jim Highsmith

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

ISBN-13: 978-0-13-359844-5
ISBN-10: 0-13-359844-6

First published, November 2013

Contents

Foreword by

Gary Gruver

Jim Highsmith, one of the key founders of the agile movement, has done us a great favor by packaging up some of his key blog posts over the last decade into an easily digestible electronic book. While a lot of the agile movement over time has focused on the details of Scrum, Jim reminds us all that agile, at its core, is about much more. It is about embracing uncertainty and developing an organization that responds quickly. It is about creating processes and approaches that are designed to teach and adjust.

The agile movement is about being able to quickly respond to the ever-evolving business environment and being able to deliver ongoing value. Jim reminds us of those facts through several different perspectives in his different blog posts. He highlights the magnitude of uncertainty facing current business leaders and provides frameworks for addressing these challenges. He shows the importance of delivering value early and often with technical approaches like continuous delivery. He also points out that the biggest challenges are not technical, but rather relate to leadership and the organization change-management process.

One of my favorite sections of this book is on organizational agility, which I found representative of what we found during my time at at HP. We started with a focus of improving the productivity of our firmware development. What we found was that the changes we implemented in the firmware development process had a much bigger impact across larger parts of the organization. It changed the value proposition we were able to provide our customers and how we managed the delivery of all the components required to deliver LaserJet printers.

Throughout these blogs, Jim challenges us all as leaders of change to remember that the agile movement is about so much more than Scrum, and we should all take a broader perspective. Please join me as we look back on his perspectives and think about how we can get better at leading our organizations through uncertainty.

—Gary Gruver
vice president of release,
quality engineering &
operations, Macys.com
San Francisco, CA

Foreword by

Christopher Murphy

I had been a follower of Jim Highsmith's writings in the blogosphere for several years when I first had the opportunity to meet Jim in person during the Agile Australia conference held in Melbourne in the second half of 2010.

During Jim's keynote address on adaptive leadership at that conference, I recall feeling particularly inspired and challenged—in the most positive sense—by his observations on leadership in dynamic and turbulent environments. For while much of Jim's thinking had derived from his experience with agile software development, I realized as I listened to the presentation that the impact was far more wide-reaching—to every level of organizational leadership.

Indeed, Jim's observations got me thinking. Change is accelerating. Why is that the case, and what is the role that technology, and technologists, are playing in that change? What does it take to not merely survive, but to actually *thrive* in an environment where change is the norm, rather than the exception? As Jim vividly demonstrates in this book, this turbulence creates enormous opportunity for those who are willing to embrace it rather than fight it. But this opportunity doesn't come automatically; on the contrary, it often requires fundamental change in leadership assumptions, models, and structures that are counterintuitive to traditional hierarchical structures and management disciplines developed during, and inherited from, the last 300 years.

It is pertinent that as I write this, I am sitting in a cottage in Derbyshire, England, in the midst of a family vacation touring the historical remains of great industries that spawned the Industrial Revolution—the significant mills, factories, canals, and mines of Northeast England. The trip has reinforced in me something that many of us implicitly know: that just as the rapid advances in 18th-century industrialization required fundamental changes in management and leadership, so do the similarly rapid technological changes of the current era. In contrast to the past, which was focused on the marshaling at scale of a largely uneducated, manual workforce, our focus now is the motivation and unleashing of the talent of highly educated, creative knowledge workers. For that, we must shed the legacies of the past and equip ourselves with a new set of leadership assumptions and models that are appropriate to the *current* revolution, not simply borrowed from the past.

For it is indeed another revolution that we are in, not only with regard to the physical hardware technologies, but perhaps more importantly for the knowledge workers who unleash their potential. The rapid increases in computing power are widely recognized and understood, and are easily demonstrated by the power of the computers that we now hold in the palms of our hands. It is less widely recognized, however, that software development—the primary method by which the advances in hardware technology are released to industry and society—has progressed in similar seismic leaps and bounds. Modern software professionals are armed with sophisticated languages, frameworks, environments, and ecosystems with which to wield their craft. That increasing sophistication is leading to an increasingly powerful ability to craft impactful software for users, in increasingly short cycle times.

Moreover, software is no longer focused on automating manual tasks—the systems of record on which so much IT effort has been expended over the last twenty years—but on software that very often *is* the business, meaning the systems that individuals, businesses, and governments use to reach and engage their users and constituents. Nimble, disruptive, and technologically driven interlopers are challenging existing business models, and entire industries, through technologically driven disintermediation. Whether traditional media, travel, real estate, publishing, retail, financial services, photography, health care, or education, the chance is that some industry is being disrupted, and that, in some way, changes in technology are contributing to that change. To understand and respond to those changes, adaptive leadership will be required.

Adaptive leadership, at its heart, is an articulation of the adaptations necessary for modern organizations to flourish in this new, technological era, and to embrace the change that it enables. As any business leader knows and will attest, and as Jim ably reinforces in this book, at the heart of any successful organization or movement is its ability to understand and articulate its own unique, ambitious mission. Jim takes this a step further: not only must that organization or movement be able to clearly articulate its mission, but it must also clearly understand the difference between responsiveness and efficiency to achieve that mission and, where responsiveness is required, implement practices around delivering a continuous stream of value by discovering and managing the flow of market and technology opportunities. It must also create the adaptive, innovative culture necessary to accomplish this. Frequently, this endeavor will involve revisiting and, as necessary, adapting or removing the vestiges of 19th/20th-century management structures such as traditional planning cycles, "command and control" management styles, hierarchy and silos, and traditional portfolio management, among others. It may sound obvious, but to be a truly *adaptive organization*, the leaders of the organization need to practice and demonstrate *adaptive leadership*.

At the organization at which I am chief strategy officer, ThoughtWorks, a global software consultancy and products business, we work for clients who are at the heart of Jim's book: organizations with ambitious missions where technology is a major enabler of competitive advantage. Many of our clients are, not surprisingly, dealing with the implications of ever-increasing turbulence and speed of change. In some instances, they are the ones creating that turbulence; they are the archetypal "disruptors" using technology to introduce a new business model or, perhaps, to disintermediate a more traditional model. In other cases, they are the more established industry leaders who are being "disrupted" and are looking to respond.

Jim's work on adaptive leadership has been of significant assistance, both to ThoughtWorks' clients and to our own business. Articulating the difference between fundamental strategies of efficiency versus responsiveness and continuous value delivery has changed our conversations with business executives, which in turn has empowered those executives to transform their organizations. Adaptive leadership has provided the language, terminology, and ideas that have helped us forge stronger partnerships with our clients, helping them to bring exciting new products and services to market in a fraction of the time they would previously have required. It has also provided guidance and ideas that have helped our own internal management development efforts, culminating in the articulation of our new market positioning bringing together our offerings and proposition into a consistent, empowering theme.

Adaptive leadership, and its underling agile principles, is no longer a set of practices for the IT department; it is a powerful business differentiator. Jim's thinking on adaptive leadership should be compulsory reading for any business leader wanting to step up to, understand, guide, and support adaptive thinking across his or her organization in the current technological era.

—Christopher Murphy
co-president and chief strategy officer, ThoughtWorks
London

Foreword by

John Crosby

As I see the pace of change continuing to increase, it's become apparent to me that organizations need to find new ways to adapt, survive, and prosper. Increasingly in our digital world, a structural competitive advantage is becoming harder and harder to maintain. Organizations that are able to embrace, experiment, and learn quickly within their environment and then take that acquired knowledge to market quickly will now succeed in the long term. As Jim highlights in this book, "discovering and executing on new opportunities is the critical capability." However, achieving this goal is not easy and requires a holistic approach within the organization. Over the last decade, I've seen the agile software delivery movement make great strides in improving the quality and reliability of software delivery—increasingly so with the advent of continuous delivery. Yet I have often seen the power of the agile approach lost when this process interacts with traditional approaches to business planning and execution. These nonagile approaches still place too much emphasis on trying to manage and inspect the inputs and deliverables as opposed to enabling great outcomes. The challenge is no longer "Can we build it?" but rather "Should we build it?"

Adaptive leadership provides a framework for individuals and organizations to address this challenge. Key to this is ensuring that at the core of the organization is the ability to instigate, encourage, and inspire disruptive thinking. At lastminute.com, we've found a culture of team empowerment critical to creating this ability. Give a team clear objectives in terms of business outcomes, be clear about constraints, and then allow the team the freedom to achieve the outcomes in the best way they identify. This may sound risky to many, and without the right approach it undoubtedly is! However, our experience suggests that this courageous leadership need not be based on blind faith. Continuous delivery has provided us with the solid bedrock of delivery quality and reliability. Paired with continuous design and the key build–measure–learn tenets of the Lean Start-up movement, it ensures we are building products that customers will both want and love.

Inevitably, there have been both challenges and learnings along the way that caused us to reevaluate our thinking. This is where adaptive leadership throughout the organization is crucial. It's not just about how the initial hypotheses of a business team may have been wrong, or how the velocity of a delivery team falls below expectations, or how either group must individually solve the problem. It's now about how

the entire organization can quickly understand and rapidly act upon this situation. For without this ongoing focus on reducing cycle time and increasing learning, you can be certain you'll be left behind by faster, nimbler, more adaptive organizations.

—John Crosby
vice president of product and technology,
lastminute.com
London

Preface

Over the past couple of years, I've been thinking, writing, and speaking about issues of adaptive/agile leadership and organizational transformations. Working with companies, working with the Agile Leadership Network Executive council, and speaking at conferences and internally with company leaders have encouraged me to focus my energy on management and leadership issues around responding to the opportunities created by today's technological and marketplace change.

The agile movement has greatly impacted software development over the last decade since the Agile Manifesto was signed. The two underlying themes of the agile movement have been reasonably successful (there's always progress to be made)—namely, building better software and increasing satisfaction (and fun) at work. In a growing number of companies, agile/Lean values and practices have been infused throughout the organization, although there remain too few of these pioneers.

As the agile movement moves past its ten-year anniversary, we need to reflect on our successes, identify areas ripe for improvements, and create a vision for the future. I've labeled the last item "agile imagineering" (paraphrasing from Disney). Just imagine what we could accomplish if more companies focused on values and quality; if they focused on achieving their ambitious missions; if they focused on increasing responsiveness to customers; if they focused on self-organizing at every level; if they focused on collaboration, transparency, courage, and technical excellence; if they focused on disruptive thinking and inspiring their people; and if they used the success generated from this transformation to promote social and economic justice in the world.

My personal ambitious mission is to inspire executives and managers to build adaptive organizations and provide them with the framework and tools to do it. Just as the original Agile Manifesto vision hasn't been fully realized in the past decade, I doubt my mission will be realized in the next ten years. However, one of the primary practices of agility is envisioning and evolving rather than planning and doing—it's about having a BHAG (Big Hairy Audacious Goal, per Jim Collins [2001] in *Good to Great*) and adapting over time to move toward that goal, rather than having a prescriptive plan and executing tasks.

This book is a compilation of papers and blogs I've written over the last several years plus some new material. The core ideas were articulated in a ThoughtWorks white

paper and subsequently expanded upon in my blog posts. One of the problems with a blog is that older entries get lost, so I wanted to pull them out into a more readily accessible format. I've rewritten material to bring a little cohesion to the disparate blogs, while other material remains as originally written.

—Jim Highsmith
Lafayette, Colorado
September 2013

About the Author

Jim Highsmith is an executive consultant at ThoughtWorks, Inc. He has more than thirty years' experience as an IT manager, product manager, project manager, consultant, and software developer.

Jim is the author of *Agile Project Management: Creating Innovative Products*, Second Edition (Addison-Wesley, 2010); *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems* (Dorset House, 2000; winner of the prestigious Jolt Award), and *Agile Software Development Ecosystems* (Addison-Wesley, 2002). Jim is the recipient of the 2005 international Stevens Award for outstanding contributions to systems development.

Jim is a coauthor of the Agile Manifesto, a founding member of The Agile Alliance, coauthor of the Declaration of Interdependence for project leaders, and cofounder and first president of the Agile Leadership Network. He has consulted with IT and product development organizations and software companies in the United States, Europe, Canada, South Africa, Australia, China, Japan, India, and New Zealand.

Chapter 1

Enterprise Agility

The agile movement has made enormous strides in the last decade, greatly improving software delivery and creating more satisfactory work environments in many organizations. The next horizon is extending agility from basic software delivery to continuous delivery and into the business itself, utilizing the advances in delivering software features early and often into a transformation of businesses to deliver complete solutions early and often. The drivers for this trend, as we will see, come from a growing focus of chief executive officers (CEOs) on trying to survive and thrive in a world of growing complexity, uncertainty, and fast-moving competition. Achieving enterprise agility, however, requires a different style of management—an adaptive leadership style. This book describes this style by considering it in three parts: (1) by describing the need for enterprise agility to engage in ambitious organizational missions; (2) by identifying what leaders need to be doing (actions) to deliver a continuous stream of value, especially as it relates to today's greatest source of change—technology; and (3) by identifying a mindset that encourages leaders to think disruptively about opportunities and respond quickly to those by creating adaptive, innovative organizations.

The core of this book is also built around two key concepts:

- Technology is the number one concern of CEOs worldwide.
- Discovering and executing on new opportunities is *the* critical organizational capability.

The first of these ideas reflects the results of a 2012 study of CEO concerns published by IBM (IBM, 2012). IBM's survey has been updated every two years since 2004, but in 2012, for the first time, technology was considered the top concern of CEOs worldwide. With mobility, social media, data analytics and big data, and cloud computing entering the lexicon of business, executives are concerned about how they can compete in this new technology-driven world. As one CEO said in the study, "The biggest risk we have is technological. If we fail to anticipate a huge technology step, we might go out of business."

In *The Upside of Turbulence*, Donald Sull makes an insightful statement that I've come to think is critical to thriving in the future: "Companies do not pass through life cycles, opportunities do" (Sull, 2009). Company life cycles are too long; by the

time company executives realize they are on the downward slope, it's often too late. Companies need to react in ever-shorter time frames; they need to take advantage of new opportunities as they arise. They need faster response and faster feedback on critical opportunities. In Sull's view, managing the flow of opportunities is even more important than innovation because it introduces a factor that many ignore when thinking about innovation—timing. Some opportunities that come along require risk taking and commitment more than innovation. This isn't to say that innovation is unimportant, but simply that applying innovation to the right opportunity at the right time is a fuller determinant of success than innovation alone.

Finally, pursuing new opportunities and bringing cutting-edge technology to your company won't be easy. For this endeavor to succeed, it needs to be driven by a big idea, a BHAG—A Big Hairy Audacious Goal (Collins, 2001). Only such an audacious mission draws people in and energizes them and provides enough driving force to overcome the trials and tribulations of disrupting your business into the future.

Managing the flow of opportunities is critical to success, and today many of those opportunities will be driven by technological innovation. Keeping abreast of new technologies, figuring out how to create disruptive uses of those technologies, and building the capability to deliver products that incorporate those technologies will be critical to success in the next decade. Keep these concepts in mind as this book's topics unfold.

CEOs and CIOs Focus on Agility

> Leaders are recognizing that our new connected era is fundamentally changing how people engage. This shift is one reason why, for the first time since this CEO Study series began in 2004, technology now tops the list of external forces impacting organizations. Above any other external factor—even the economy—CEOs expect technology to drive the most change in their organizations over the next three to five years. (*Leading Through Connections: Insights from the Global Chief Executive Officer Study* (2012). Reprint Courtesy of International Business Machines Corporation, © 2012 International Business Machines Corporation.)

Enterprise agility may be at a tipping point, much like agile delivery was in 2001. In 2010 and again in 2012, IBM interviewed more than 1500 CEOs and published in-depth studies of its findings. The company's 2010 study "Capitalizing on Complexity" focused on what CEOs saw as the marketplace challenges and the key strategies for surviving and thriving.

"Our interviews revealed that CEOs are now confronted with a 'complexity gap' that poses a bigger challenge than any factor we've measured in eight years of CEO research." As this quote from the 2012 study indicates, technology is now at the head of that complexity gap. In 2004, 2006, 2008, and 2010, market factors topped the list of CEO concerns, while the 2012 study illustrates the rising criticality of technology to company futures. Technology is the number one disrupter, while at the same time it is the number one way to thrive from that disruption. Eight in ten CEOs expect their environment to grow significantly more complex, and fewer than half believe they know how to deal with it successfully (IBM, 2010).

The CEOs cited three key factors to success in this turbulent environment:

- Embody creative leadership
- Build operating dexterity
- Reinvent customer relationships

Creative leadership includes embracing ambiguity, taking risks that disrupt the status quo, instituting new management styles, and making decisions more quickly. Building operating dexterity requires simplifying whenever possible, managing systemic complexity (standardization in some cases), and promoting a fast and flexible mindset. Reinventing customer relationships involves honoring customers, using two-way communications (increasing the use of business social networking), and profiting from the information explosion (taking advantage of new data analytics).

"To create a profile of dexterous organizations, we grouped those CEOs who recognized the value of fast decisions, an iterative approach to strategy and the ability to execute with speed" (IBM, 2010). One of the key questions looking at these factors is "How do we train people to do these things?" The answer, in part, is by including "traits" in product development planning. In *Connecting the Dots*, Harvard Business School professor Warren McFarlan and consultant Cathleen Benko (McFarlan & Benko, 2003) use three criteria to prioritize work on products (or projects): short-term objectives, long-term objectives, and trait objectives. The last of these, trait objectives, deals not with capabilities but rather with mindset or mental models. They speak to the future and the personal and managerial mental models that will power organizations into the future. Adaptive leadership is more about altering mental models than about implementing practices and processes, but practices and processes certainly instantiate those mental models. Agility is a trait objective that will help organizations respond to business complexity and complication. Agile delivery projects and their impact on the wider organization can fuel the mental model changes required to build more dexterous organizations.

WHY AGILITY?

> To effectively respond to these new dynamics (the digital world), companies must begin thinking about ways to broaden their ecosystems and revenue streams while becoming more responsive and agile. (McKinsey & Company, "Competing in a Digital World: Four Lessons from the Software Industry" [Sikes, 2013])

One of the "lessons" articulated in this article in the McKinsey quarterly is the need to create an agile organization, because this step is required for the other three lessons to succeed. At the core of an agile, or adaptive, organization is the ability to instigate, encourage, and inspire disruptive thinking. Surely adaptive leadership is about learning, adapting, collaboration, and more—but disruptive thinking lies at the core of these behaviors. Disruptive thinkers see the same information others do, but they interpret it differently. Disruptive thinkers aren't satisfied with the status quo, or even with incremental improvements; they are looking for big wins. Disruptive thinkers challenge your mental models about the world, and they challenge their own as well. Much has been written about disruptive change—in technology, in business models, in product development speed, in politics and economics—but finding, keeping, and encouraging disruptive thinkers in your organization will be hard—very hard. The demise of company after company, some at one time at the cutting edge of their industries, attest to the difficulty.

BUSINESS AGILITY NEEDED IN TURBULENT TIMES

Agility is a business imperative, not just a technological one. Agile software development has enjoyed great success over the past 10 years and agile project management has made inroads into the project management community, but there is still a long way to go. Many companies relegate agile methods to the position of "just another in a long line of software engineering techniques," while in others the transition to agile methods stalls after a few projects, even though those projects are successful. Too few agile transitions make an impact outside of software delivery groups.

What is missing? The agile movement has the potential to be strategic to businesses, particularly those whose overall strategy focuses on responsiveness over efficiency. We are selling ourselves short! We have the potential to energize new business models, engage middle and upper management in becoming agile, and change the way product and project managers connect agile concepts and practices with upper management.

"How in the age of rapid change do you create organizations that are as adaptable and resilient as they are focused and efficient?" (Hamel, 2009). Two factors that create

turbulence in business environments are complication and complexity. Something that is complicated has many parts and is hard to understand. Something that is complex is unpredictable (or at least less predictable). A Boeing 767 is complicated (many parts) but not complex. However, the process of designing a 767 would be both complex and complicated, as many uncertainties would arise during this design process. Complexity can be further defined by volatility (speed, magnitude) and ambiguity (the haziness of reality), as well as by uncertainty (lack of predictability). But turbulence has an upside—creating opportunities for those who can weather the dynamism and complexity.

RESPONSIVENESS AND EFFICIENCY

The previously cited IBM CEO study focused on the strategic issues of complexity and complication. Whether the name assigned to the approach used to tackle these problems is agility, adaptability, dexterity, or responsiveness, the capability is critical to success. For the purposes here I'll use the term "responsiveness" and call the opposite strategy "efficiency." While companies may strive to achieve both, one of these factors must be the driver, the objective, and the other a constraint. Constraints help guide companies, but they are different from objectives and the two shouldn't be confused. Responsiveness is a business strategy. Agility and adaptability achieve that strategy.

As an example, consider Google and Wal-Mart—both very successful companies, but each coming from a different perspective. Google focuses on responsiveness, creating new products at a prodigious rate. Does Google's management have an interest in efficiency? Of course, but its goal of responsiveness and efficiency is a constraint. Wal-Mart is the opposite. Obviously, Google's strength and competitive edge comes from its responsiveness: it is able to create new service offerings quickly, to deploy new features early and often, and to take advantage of new opportunities quickly. Constraints keep costs within reason, but they are not objectives. In contrast, Wal-Mart's strategy focuses on efficiency and driving costs down in every way possible. Does Wal-Mart need responsiveness in some of its operations? Of course, but it doesn't confuse which is the objective and which is the constraint.

Not all businesses need the same degree of responsiveness, but improving response can often pay big dividends. A large retail company recently was able to add a major new brand to its offerings in a matter of a few months—impressive, but not extraordinary, for a new online presence. However, the next step was extraordinary; the IT organization—the bits and bytes people, not the bricks and mortar ones—built the first actual store. Moreover, they accomplished that feat many months sooner than estimated by the bricks and mortar department! The IT group's agility mindset and practices extended beyond software development: they prototyped, they took

shortcuts (not hooking up to all corporate systems, for example), they improvised—but they got the store up and running fast and allowed the company to confirm a decision to build additional stores.

> **Agility Generates 30% Higher Profits**
>
> An overwhelming majority of executives (88%) cite organizational agility as key to global success. Other studies support this idea as well: research conducted at MIT suggests that agile firms grow revenue 37% faster and generate 30% higher profits than non-agile [firms]. Yet most companies admit they are not flexible enough to compete successfully. Internal barriers stall agile change efforts. The main obstacles to improved business responsiveness are slow decision-making, conflicting departmental goals and priorities, risk-averse cultures and silo-based information. Technology can play an important supporting role in enabling organizations to become more agile companies. (Economist Intelligence Unit, *Special Report on Agility*, March 2009)

For a number of organizations, the driving force for their agile/Lean transformations will be the ambitious mission of improving their responsiveness to the flow of opportunities that arrive every day on their doorstep.

How Agile Do We Need to Be?

Organizational agility is the ability to respond to opportunities in the marketplace, or as Donald Sull defines it, "the capacity to identify and capture opportunities more quickly than rivals do." The follow-up question should be, "How agile or responsive do we need to be?" This last question is one that organizations rarely ask themselves, or at least they don't ask it in enough depth. What follows are six different ways in which this depth of needed analysis can be accomplished.

The first area for analysis is organizational units. For starters, some organizations are conglomerates of multiple companies. Companies then may be organized along business units that may be product type oriented, plus a separate shared services business unit. Other layers of organization may be functional areas (marketing, sales, manufacturing, and product development). The "How responsive do we need to be?" question needs to be asked at every level. One product may be more mature or market dominant than another. Marketing may need to be more nimble than finance. Product A may be competing in a more volatile market than Product B.

A second area for analysis is the type of change anticipated—disruptive, differentiating, or incremental. Incremental change includes small new items added to an existing product, practice, or process. They are the small "ah ha" tweakings that improve products. Differentiating changes involve a significant product, practice, or process

enhancement that creates a clear differentiation from the competition and that keeps the company ahead of its competition for some short-to-medium time frame. This type of change creates differentiation within a market, but doesn't significantly change a market. Disruptive change requires innovating to create an entirely new market or business model. Disruptive changes can drastically impact companies or even industries (for more detail on these types of change, see the section "Change Isn't Change" in Chapter 4).

Based on the type of changes anticipated, the capabilities required to respond are different as identified by Donald Sull in *The Upside of Turbulence*:

- Strategic: spotting and seizing game-changing opportunities
- Portfolio: shifting resources—including cash, talent, and managerial attention—quickly and effectively out of less promising business areas and into more attractive ones
- Operational: exploiting opportunities within a specific business model

A third area for analysis is deciding how to manage the flow of opportunities. Opportunities go through four stages—identification (scanning for possibilities), start-up (expanding the opportunity into a minimum viable product with validated customers and business model), scaling (ramping up to volume), and maturing (managing the maturing and decline). For example, Lean Start-up concepts and practices might be very useful for a new product or a new business opportunity. Regularly scanning the technology space (see, for example, the ThoughtWorks technology radar) could alert an organization to an opportunity for experimentation.

A fourth area for analysis, and one related to the type of change anticipated, is what changes in response—the product, product features, customers, technology, or business model. A disruptive change might create a strategic response that introduces a new product. A differentiating change might require an organization to reallocate money and talent from one product line to another. A competitor's incremental product change might require a quick response to match its new features. Part of this analysis should include looking at cycle/lead times for technology or product results—how often new products/features need to be deployed and how much lead time is required for new products/features (development time from backlog release to deployment).

A fifth area for analysis would be an organization's impediments and enablers—technology, practices and processes, capabilities, and culture. Continuing the last example, introducing a new product quickly might require a significant change in technology and continuous delivery capabilities. Rapid reallocation of assets might require a change in portfolio management practices and a cultural change. Building a capability to deploy new features quickly (think continuous delivery) might require changes in functional unit processes.

A final area for analysis might be the "continuous flow" requirements for organizational units or products. Both lead time and deployment frequency should be identified. Lead time (time for a feature to move from backlog release to deployment) and deployment frequency (monthly, weekly, daily, multiple times per day) requirements help an organization determine which capabilities it needs in this area.

Few organizations appear to do the depth of analysis and strategizing about which "level" of agility makes sense for them. They talk about improving delivery speed, increasing responsiveness to customers, reducing costs, or improving quality—but they don't take the next step and describe the benefits in a compelling manner. They don't analyze and prioritize desired benefits—they want them all, not understanding that there are differences in how one would implement agile methods to reduce costs versus to increase responsiveness.

MANAGE THE FLOW OF OPPORTUNITIES

The critical reason companies need to manage opportunity flow, rather than company life cycles, is the speed of change. Company life cycles are a combination of many opportunity life cycles, but by the time problems with the aggregate are recognized, it may be too late. The problem today is that opportunities bombard us, and sorting out which ones to pursue and which ones to pass on is a never-ending struggle. Success comes from having an ambitious, well-articulated mission; a flare for correct timing; a resolute commitment to an often risky path forward; a collaborative, innovative team; and, of course, a bit of luck.

The life-cycle flow described in this section incorporates phases (and processes within those), but my use of the word "flow" shouldn't be visualized as indicating a product funnel. It's not a meandering flow through an ever-narrowing boundary, but more like a raging torrent that could sweep you away at any moment. Opportunities could spring up quickly and leave just as fast. This section (and the next on organization) will attempt to answer three key questions:

- Which types of opportunities might we encounter?
- What are the phases that opportunities move through over time?
- Which alternative organizational structures might we use?

Opportunity Types

Before identifying opportunity types, and because innovation is a big part of taking advantage of opportunities, it might be helpful to define innovation. This proves to be much easier said than done. A short Google search turns up more than 50 definitions of innovation. I've cobbled together a definition that makes sense to me: "Innovation

creates something substantially different—a process, product, or service—that generates new value." Many definitions of innovation mention "different" (or "new") and "value" but don't use adjectives. Is a different (or even a new) paint color on a product innovative? To me, some adjective such as "substantially" is needed to qualify "different." Similarly, the result—economic or social value—should have a new component to it and not just be a little more of the same.

The three types of changes or opportunity types are incremental, differentiating, and disruptive, as described in the previous section on how agile we need to be.

Phases

The phases in the flow of opportunities are shown in Figure 1.1. This figure is based on one in Don Sull's work with a couple of additions. The five phases are identified here:

- Discovery: Find good opportunities (my addition)
- Start-up: Confirm a viable customer base, business model, and product
- Scale: Commit major resources to fulfill the mission
- Mature: Avoid sense of lock-in
- Decline: Limit new investment

The second difference from Sull's diagram is the addition of an opportunity type to each of the entries (bubbles). In the early Discovery and Start-up phases, it is particularly important to determine whether the opportunity is differentiating or potentially

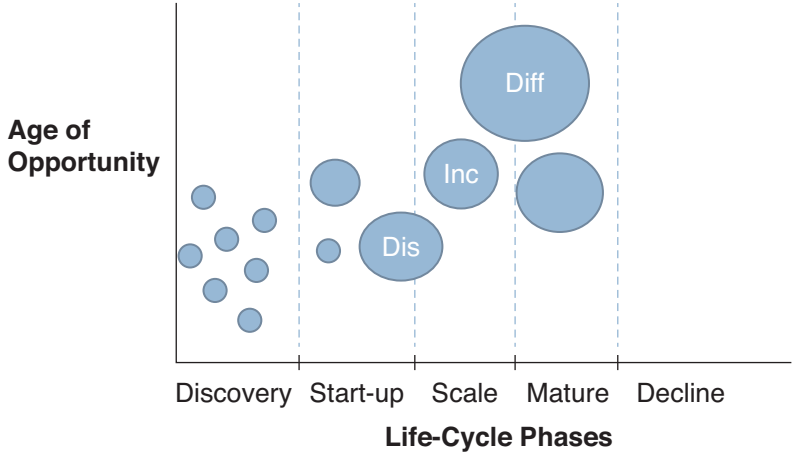

Figure 1.1 The Flow of Opportunities (Adapted from Sull)

disruptive (I say "potentially," because identifying disruptive opportunities early is exceedingly difficult). (Note: The sizes of the bubbles in the diagram indicate the magnitude of the investment required.)

Discovery

While Don Sull's opportunity phases begin with Start-up, I think there is a step prior to that—Discovery. Opportunities abound today. They come in many forms—technology, business model, social action, management practices, new products, and more. The Discovery phase focuses on finding "good" opportunities out of the myriad of possibilities. Not every discovery deserves a Start-up experiment.

One visual approach to displaying the spectrum of possibilities is the opportunity radar shown in Figure 1.2. Organizations should customize the quadrants in the radar starting with management practices and social action, technology, products and services, and business models. Each entry in the radar would be characterized as Hold (proceed with caution), Assess (worth exploring with the goal of finding out how it will affect you), Trial (worth pursuing; start with a low-risk project), or Adopt (industry as a whole has finished trial; patterns for usage established). Each entry is also typed (incremental, differentiating, and disruptive). Over time, both

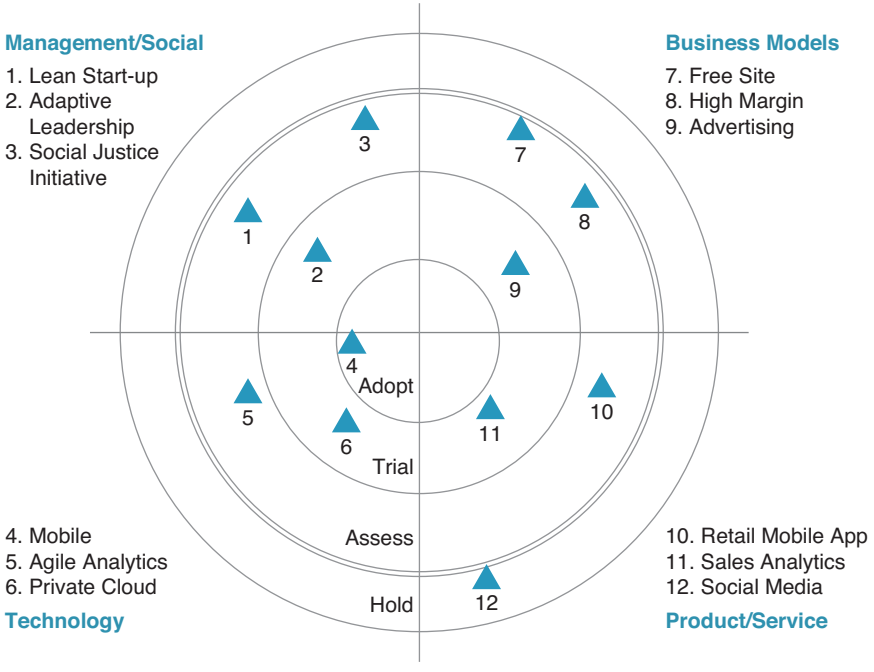

Figure 1.2 The Opportunity Radar

characterizations and types might change. While we talk about the flow of opportunities, the progression should not be viewed as a traditional pipeline, narrowing choices at each phase. Opportunities are more volatile than that—for example, moving from Assess to Trial status, and then back as uncertainty changes.

The opportunity radar operates in conjunction with the phase diagram—at least in the Discovery, Start-up, and Scale phases. (Note: Based on ThoughtWorks' Technology Radar.)

Start-up

The objective of the Start-up phase is to reduce risk and uncertainty around the customer, the business model, and the product. New companies may pursue a single start-up opportunity, while established companies may have a portfolio of them. Just as continuous development and continuous delivery have become part of agile and Lean methods, so continuous discovery—the front-end piece of defining products, markets, and business models—has evolved over the recent past. Much of this front-end work has emerged from the Lean Start-up movement, characterized in Eric Ries's 2011 book, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Whereas agile/Lean product development focuses on building products quickly and effectively, Lean Start-up focuses on making sure there are markets, customers, and sustainable business models for those products.

Lean Start-up and agile delivery make a powerful combination to convert opportunities into new products and businesses in a fast-paced, evolutionary manner.

Scale

Scale is where opportunities go from little businesses to big businesses. If Discovery and Start-up are about finding and validating opportunities, Scale is about fulfilling potential. Also, while the first two phases require entrepreneurial spirit, the latter requires executive courage. The core issue is summed up by Geoffrey Moore (2011): "You have to escape the gravitational field of last year's operating plan."

Successful scaling has eluded many companies. While it might not seem as exciting as start-up, scaling is equally difficult, just in different ways. Scaling usually requires a big commitment in dollars and talent—in other words, executive and management courage. Finding a few million dollars to spend on a start-up opportunity might be relatively easy, but finding $50 million to scale up that effort probably requires cannibalizing some other part of the budget. Very few managers want to lose "their" budget—an extremely strong "gravitational" field, in Moore's words. He adds, "The transition from invention to broadly scaled deployment is a race against time to get past the tipping point that separates the hits from the flops" (Moore, 2011).

In addition to shifting resources, successful scaling may require significant organizational changes as described in the next section.

Mature and Decline

3M has long had a corporate vision that 30% of the company's revenues will come from products that are less than 3 years old. Top-level visions like this make it easier (but not easy) to manage the maturity and decline of products or markets. They require senior executives to be realistic about "category" power. As Moore (2011) says, "Category power, in short, is the number one predictor of future economic performance." Print media versus digital media is a category shift that has provided both opportunity and peril for many companies. Some companies have successfully made the transition by shifting internal resources from print to digital, while others have made an external shift through bankruptcy. Kodak, for example, tried to make the shift from film to digital, but never achieved the right balance between scaling up its digital photography efforts while managing the decline of print film.

Effective management of the Mature and Decline phases is critical to free up resources to fund the start-up and scaling phases. It is a delicate balancing act because the revenue and profit stream from mature products has proven very difficult to give up. In addition, especially with disruptive opportunities, the cost structure of new products may be drastically lower than with the mature product. Asking mature organizations to drastically reduce their cost structures can be a daunting prospect. (For more on the issue of disruptive opportunities, see the classic in the field, *The Innovator's Dilemma*, by Clayton Christensen [1997]).

ORGANIZING FOR INNOVATION

One of the questions I get most frequently goes something like this: "Should we create a separate agile group, team, department, or product team, or should we plan our agile transformation within our current organization?" This question arises not only for agile transformations, but also for opportunity management and innovation in general. Many people approach this question as a problem to be solved once and for all; in truth, the question is a classic paradox that has a resolution, not a solution. Resolutions are temporary, not permanent, and change from time to time as the situation changes.

Four criteria need to be analyzed in coming to an organizational resolution, and four basic organizational structures are possible. The four criteria are change type, phase of development, opportunity size, and culture. The four organizational structures are in situ (current), stand-alone (skunk works), ambidextrous, and dual operating systems.

Criteria

The three types of changes—incremental, differentiating, and disruptive—were described in a previous section, as were the five opportunity phases—Discovery, Start-up, Scale, Mature, and Decline.

The size of the change or opportunity also impacts structure. A small agile project, for example, could be launched with minimum organizational impact, whereas a 500-person project would require more analysis and probably greater disruption.

Finally, culture influences how any change is approached. Some companies have carried off a large agile transformation across the entire organization at one time, whereas others would have found (and some have found) this approach to be disastrous.

Organizational Structures

The first organizational structure for change is to use the existing structure, keeping things as they are—that is, in situ. This option might work well for an incremental change, but it is less attractive for a disruptive one, as disruptive changes are usually large (in potential at least) and call for cultural changes.

The second structure has been used extensively—a skunk works that is a stand-alone organization. This type of structure has been used as organizations have moved to the Web, with digital presence development groups splitting off from central IT organizations. In the early stages, these groups are often product-centric. New products, particularly innovative ones, often suffer when they remain within an existing organization, so new groups are set up. The problem with skunk works is that they are often separate from the main organization and have less political influence because they report to a relatively low level in the main organizations. The most famous skunk works, which came up with innovation after innovation in the 1980s, but whose innovations were not translated into commercial success, was Xerox Parc.

The differences between a skunk works and an ambidextrous[1] structure are basically reporting level and the presence of all business functions. Skunk works are usually product related, but they are not complete businesses with their own marketing, sales, manufacturing, and other support functions. When the separate group has all those functions, it usually reports to a high executive level. One very successful such organization was the original IBM PC group. This organization was completely self-contained (e.g., sales, manufacturing, product development) and was physically

1. Also see the section on ambidextrous organizations in Chapter 2.

separated from IBM headquarters in New York. The parent organization had a very different culture and a different approach to development (partnering with Microsoft was one such deviation) from the ambidextrous PC group (the IBM PC group was located in Florida).

The downside of ambidextrous organizations is that the changes they make don't get promulgated to the rest of the company. Many financial and retail companies have set up separate digital presence groups that have innovative, fast-moving cultures—but these changes don't permeate the rest of the company, or do so only very slowly. Although some types of changes don't need to affect the wider organization, for others broader distribution would be advantageous.

Finally, the newest twist on organization structure comes from John Kotter's (2012) *Harvard Business Review* article, titled "Accelerate! How the Most Innovative Companies Capitalize on Today's Rapid-Fire Strategic Challenges—and Still Make Their Numbers." Kotter, who has written several well-known books on change management, says:

> Perhaps the greatest challenge business leaders face today is how to stay competitive amid constant turbulence and disruption. The existing structures and processes that together form an organization's operating system need an additional element to address the challenges produced by mounting complexity and rapid change. The solution is a second operating system, devoted to the design and implementation of strategy, which uses an agile, networklike structure and a very different set of processes. I'm proposing two systems that operate in concert.

Kotter's dual operating system begins with the existing organization—one that handles current business operations well. He then superimposes a network structure on top of the existing structure. Kotter's network system has some additional hallmarks:

- It is called a "Guiding Coalition" to differentiate it from the traditional hierarchical structure.
- Individuals "Volunteer" to be part of this structure.
- The structure operates as a "Self-organizing" team.
- The structure contains "Many change agents" who are part of the management structure rather than a separate team.

The network structure is responsible for new strategies, innovative products, and other forward-thinking projects. It leads the way to the future. However, the people are the same. It's not that one group thinks about strategy and another ponders execution, but rather that the same individuals do both.

THE STRATEGIC IMPACT OF CONTINUOUS DELIVERY

Continuous delivery is one of the exciting new trends in software development. According to Humble and Farley (2011), the purpose of the practices and principles of continuous delivery is to encourage "greater collaboration between everyone involved in software delivery in order to release valuable software faster and more reliably." Continuous delivery is an extension of the agile practices that deliver value to customers early and often. As shown in Figure 1.3, it combines continuous design and delivery: a front end of customer experience and technical design in an iterative process, continuous integration (characterized by comprehensive automated testing), and continuous delivery (the ability to deploy new releases to production frequently). This enterprise value creation model extends over the entire application development life cycle—from inception to deployment, with high levels of automation throughout.

Jez Humble says that there are three strands to continuous delivery: one strand concerned with automation of build, test, deployment, database migrations, and infrastructure; a second strand concerned with practices, such as continuous integration, good configuration management, and testing; and a third strand concerned with people, having everyone involved work together throughout the software delivery life cycle. While these appear to be technical issues, continuous delivery involves critical organizational collaboration (development and operations, for example) and business process changes.

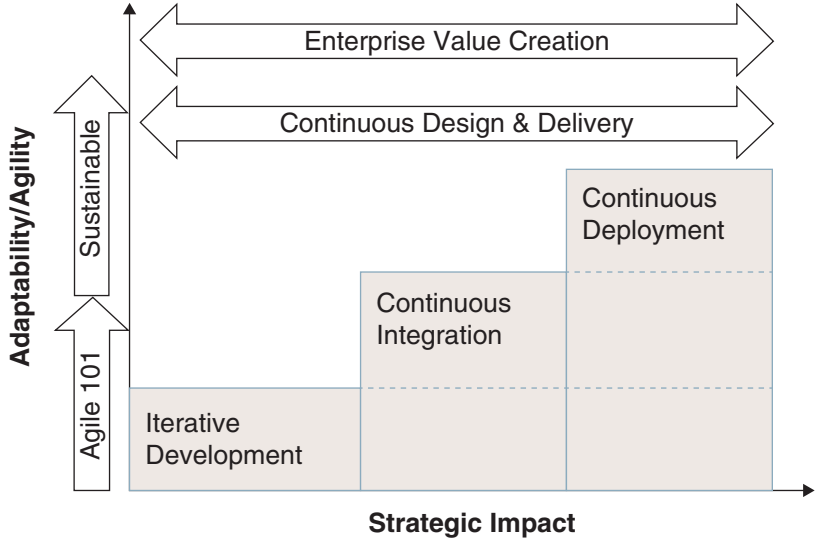

Figure 1.3 Continuous Delivery

Flickr releases new changes to its website multiple times per day. How would the capability to release new features—monthly, weekly, daily, hourly—impact your business? If your business is SaaS (software as a service), the impact could be strategic, but every business can benefit from fast and flexible releases in some way. The real questions revolve around business process changes and management's ability to find innovative ways to take advantage of continuous delivery.

Gaining business value from continuous design and delivery is associated with two other issues: strategic impact and agility. First, as a company progresses farther to the right on the horizontal axis of Figure 1.3, additional investment and organizational collaboration are required. Therefore, from a business value perspective, companies need to assess the strategic impact of the progression. While continuous delivery can reduce cost and risk (through more automation), the most significant benefits arise from frequent release of new software functionality. The key question then becomes, "How can we benefit from releasing new functionality monthly, weekly, or even daily?" Furthermore, "How will our organization and business processes need to change?"

In large organizations, IT applications support a variety of business areas. For some applications, the benefits of continuous delivery may be enhanced revenues; for others, the benefits may take the form of cost and risk reduction. In thinking about implementing continuous delivery across an application portfolio, companies should begin with those areas that have the biggest strategic impact, those applications with revenue-enhancing prospects.

The second issue in realizing the benefits of continuous delivery is the organization's agility or adaptive maturity. Many organizations seem to be stuck at Agile 101, the rule-based approach to the agile world (do this, don't do that). This is a necessary first step toward becoming agile, but it's only a first step. To take advantage of the fast-paced responsiveness of a continuous delivery environment, the entire organization—from delivery teams to executive management—needs to embrace the process changes required to respond rapidly, collaborate effectively between development and operations, and adopt an adaptive, exploratory mindset.

Continuous delivery has the potential to change the competitive landscape, but only for those companies bold enough to take enterprise agility seriously. It may be the next big step in delivering strategic business value to clients quickly, with lower risk and possibly lower cost. However, it won't happen unless managers understand its potential strategic impact and the enterprise agility and adaptive leadership necessary to implement it.

LEVELS OF AGILITY: STRATEGIC, PORTFOLIO, OPERATIONAL

One reason why organizations have difficulty implementing agile and Lean methods is that they fail to tie the reasons for this implementation to their business strategy.

> Agility is the ability to both create and respond to change in order to profit in a turbulent business environment. (Highsmith, 2009)

The degree of agility required by the business then indicates which level of agile transformation may be viable.

Figure 1.4 shows three levels of agility that organizations may strive to achieve—operational, portfolio, and strategic. Organizations need to be very clear about which level they aspire to and whether that level corresponds both to their business strategy and to the benefits they want to achieve.

The operational level focuses on improving the delivery of software projects. Regardless of the long-term goals, every effective agile transition begins at this level. Software delivery projects are enhanced, but no wider organizational change takes place. If there isn't a driving business need for responsiveness, then operational agility may be all that is reasonably achievable. There will, therefore, always be tension between delivery staffs who are agile and project, product, and line managers who operate as they did before the introduction of agile practices.

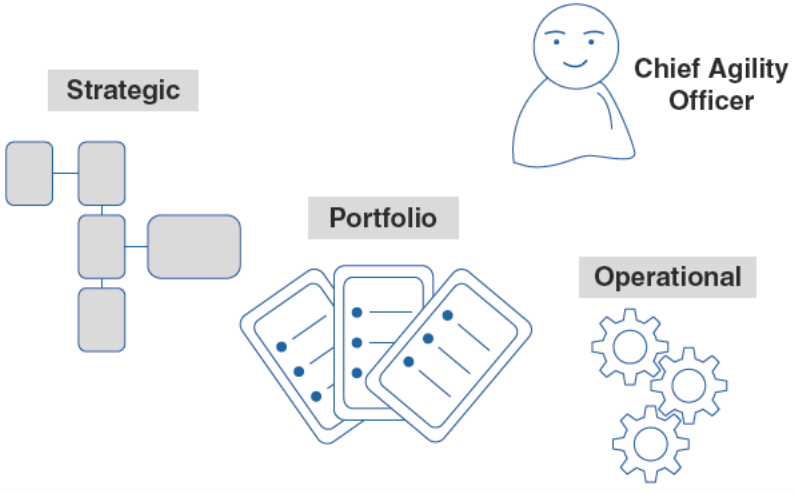

Figure 1.4 Levels of Agility

The strategic level focuses on achieving responsiveness throughout the organization: within IT, in other functional areas, and spreading up into management and leadership positions. A number of software companies and a few other firms have achieved this strategic level of agility.

At the portfolio level, agile practices have moved up somewhat in the IT or software development organizations, but have not been fully embraced at the top and haven't spread outside IT. Projects that have certain characteristics are slated for agile development, but traditional development projects persist within the company. The portfolio level actually holds tremendous opportunity—to focus on high-value projects, to radically reduce the scope of projects, to reduce work-in-process bottlenecks, and to reduce the amount of time spent doing project analysis and estimates. Agile portfolio management is the bridge between enterprise agility and successful execution of agile projects.

While organizations can move from operational to strategic agility over time, there isn't a right level—only a level that matches an organization's responsiveness strategy and business goals.

THE CHALLENGE OF ADAPTIVE LEADERSHIP

Leaders often forget or don't understand the difficulty a company's staff has in transitioning to an agile delivery model. Programmers have to change how they do testing (a technical change) and how they interact more collaboratively with others (a social change). Product managers have to change their interactions with delivery teams—increasing their availability, managing backlogs, and engaging with the delivery team. These are often challenging changes.

Adaptive leadership is the work of energizing, empowering, and enabling teams to rapidly and reliably deliver business value by engaging customers and continuously learning and adapting to a volatile environment. What many agilists, or their executives and managers, haven't realized is that the changes that leaders face are just as wrenching. Leaders face the same two challenges as delivery teams: doing different things and behaving differently. The latter entails changing their mental models about how best to improve performance. For example, here are just a few of the things adaptive leaders need to do:

- Create an agile performance management system
- Align agile transformation efforts to business strategy
- Determine operational, portfolio, and strategic agility aims
- Facilitate a decentralized, empowered, collaborative workplace
- Foster adaptable IT, product line, and product architectures
- Create an agile proficiency evaluation framework

This list could be greatly expanded upon, as could one that listed adaptive mindset characteristics. However, this book concentrates on the critical aspects of being agile and doing agile that managers and executives need to focus on first—that is, the most critical aspects of being adaptive leaders. These issues are outlined in Figure 1.5. Note that this model is a starting place, a core on which to expand. Just as asking programmers to collaborate for the first time may be difficult for them, so tackling the tasks in this model may be difficult for leaders. "Doing less," for example, isn't usually in managers' lexicon— they want to do more and more. In reality, figuring out the primary focus and eliminating marginal and low-value work can bring substantial benefits. If these things were easy, they probably would not be worth doing. Growing adaptive organizations requires managers and executives who can lead, who can take risks, who can seize opportunities, and who ultimately are courageous.

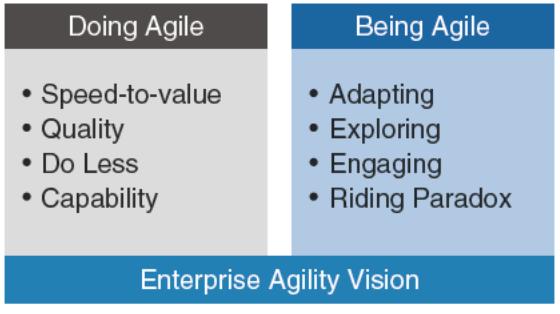

Figure 1.5 Enterprise Agility Vision

Chapter 2

Adaptive Leadership Today

This chapter addresses leadership issues that range from what leaders want from agile today, to the future of the agile/Lean movement, to integrating economics, product, and social responsibility.

Creating a Culture of Adaptation and Learning

> When outcomes are uncertain, answers hard to devise, That's the time to form a team, tap dreams, and improvise.... Putting lipstick on a bulldog won't transform enough, Makeup can't hide everything; change takes deeper stuff. (Kanter, 1983)

In *e-volve!*, Rosabeth Moss Kanter sends two clear messages: there is a new digital culture of the future that we have to understand, and a little makeup here and there won't be enough to meet the challenge. Her words are still true—three decades later. A little less documentation, sprinkling an "extreme" practice here and there, revamping a couple of procedures, or pseudo-empowering teams won't be enough to build the organizational ecology required to deliver value as markets change again and again.

The new digital frontier "is a frontier of technologies, ideas, and values," says Tom Petzinger, author of *The New Pioneers* (1999):

> The new pioneers celebrate individuality over conformity among their employees and customers alike. They deploy technology to distribute rather than consolidate authority and creativity. They compete through resilience instead of resistance, through adaptation instead of control.
>
> Indeed when you look deeply enough into a growing organization or spend enough time with thriving entrepreneurs, business begins to look very much like pure biological evolution, propelled into real time on the fuel of human intelligence.

The central question related to turbulence, change, and our changing social systems is this: do our traditional, fundamental models of the world and how to manage make sense anymore? For a growing cadre of scientists, business executives, and agilists, the answer is no.

Since the mid-1980s, scientists have explored the concepts of complex adaptive systems (CAS) in increasing depth. Similarly, since the mid-1990s, a growing group of managers began applying CAS concepts to make sense of how organizations operate in an increasingly complex world (for the latest rendition of applying CAS to management, see *Management 3.0* by Jurgen Appelo [2011]). Many, but not all, of the leaders in the agile movement have based their ideas about management on the sense-making ideas of complex adaptive systems. When Kent Beck talks about "organic design" or "generative rules," his conceptual foundation is an understanding of complexity, as are Ken Schwaber's ideas when he designs daily Scrum meetings.

Five key ideas about complex systems are helpful in understanding organizations in new ways. These ideas reinforce ideas that managers have found to work in actual practice.

- Complex systems, be they biological or human, are composed of independent, decentralized agents.
- These independent agents, in the absence of centralized control, will self-organize.
- Self-organization will create complex behavior and emergent results that are not evident from studying the agents themselves.
- Rich information flows in an ecosystem balanced at the edge of chaos define the most effective pattern for generating emergent results.
- Simple, generative rules guide the creation of complex behaviors.

The Agile Manifesto principle that individuals and interactions are more important than process and tools is a reflection of these complexity principles. Individuals are independent, and it is the interaction between those individuals that creates innovative (emergent) results. If these agents are guided by a simple set of rules (the Agile Manifesto, for example) rather than volumes of processes and procedures, they have a much better chance of responding to the complexity of the world around them. Why? In a turbulent, oft-changing environment, a set of comprehensive procedures will never be comprehensive enough to address every situation, and the more comprehensive they become, the more difficult they are to apply. Simple rules guide innovative, intelligent responses; comprehensive rules guide rote, routine responses.

In complex, uncertain, ambiguous environments, adaptation is significantly more important than efficiency. Efficient practices are based on assumptions of predictability and control. If the environment is relatively predictable, and the problems at hand are similar to ones we have encountered before, then efficiency practices—getting better and better at what we already know how to do—are useful. In contrast, in unpredictable environments, those in which we have little experience, maintaining the flexibility to try different approaches, making mistakes and learning from them, and embracing change rather than fighting it become more useful strategies. Today,

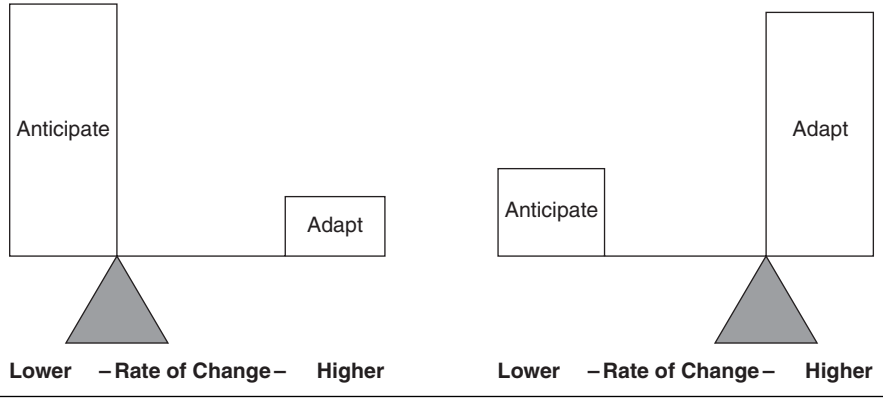

Figure 2.1 Balancing Anticipation and Adaptation

organizations require more of the former (adaptive practices) and less of the latter (efficiency practices), as shown in Figure 2.1.

Rich information flows in an ecosystem balanced at the edge of chaos define the most effective pattern for generating emergent results. Emergent results, creativity, and innovation require a rich stew of information, of conversations and interactions. The "edge of chaos" defines a narrow band in which there is enough order to stave off randomness, yet enough disorder to generate new ideas. This balance point might be referred to as the zone of creative adaptability. Dee Hock (1999), in *Birth of the Chaordic Age,* refers to "chaordic" management, meaning balancing between chaos and order. Creating an adaptive organization involves balancing on this precipice—a very uncomfortable position for those raised within a management culture whose worldview is one of Newtonian certainty.

WHAT DO LEADERS WANT FROM AGILE?

I've just returned from a busy 4-week trip to first Germany and Australia, and then Chicago. In shaking off the jet-lag cobwebs, I took some time to reflect on this trip and others over the last 6 to 8 months (the United Kingdom, Brazil, and the Agile Executive Summit). On these trips I've talked with many executives and managers who are attempting to transform their organizations in one way or another. Issues that keep coming up appear to be somewhat universal across the world. Leaders and managers share the following characteristics:

- They are eager to understand their role beyond software delivery.
- They want to explore how to expand agile concepts outside IT.

- They are figuring out ways to be more responsive to business and technology opportunities.
- They are eager to connect with other leaders and managers.

First, the agile community has been focused, and rightfully so, on delivery issues—the principles and practices that have improved software delivery in many organizations. From small projects in the early years, agile teams have expanded their repertoire to include large projects, distributed projects, projects in regulated industries, business intelligence projects, and much more. However, less emphasis has been given to what managers and leaders need to do to further the goal of delivering a continuous stream of value from their development organizations. The agile and Lean communities are beginning to address these issues in a more substantive way, but there is a long way to go and many managers would like more ideas and direction.

Second, leaders and managers are increasingly asking how to influence their organizations in an agile/Lean manner beyond the boundaries of IT. These forward-thinking individuals recognize that to take full advantage of agile IT, the enterprise outside of IT needs to be more fully engaged in responding rapidly to environmental opportunities. Furthermore, this movement toward enterprise agility is being driven not just from internal IT, but also from management trends such as those described in *Radical Management* (Denning, 2010), *Beyond Budgeting* (Boqsnes, 2008), and *Lean Startup* (Ries, 2011).

Third, leaders and managers are looking for integrated ways to improve their responsiveness to business and technology opportunities. This actually involves the integration of enhanced delivery practices and wider use of agile practices in enterprises to dramatically reduce cycle times using practices like those found in continuous delivery. Cycle time reductions that enable better business responsiveness come in two flavors—feature delivery cycle time and release frequency. The former is the time it takes for a feature to traverse the delivery pipeline from initiation to release, and the latter is the frequency of actual release (e.g., daily, weekly). IT executives are searching for ways to increase responsiveness in areas such as mobility, cloud computing, and Big Data, while maintaining efficiency in their core applications operations.

Fourth, managers and leaders are eager to connect with their counterparts in other companies to discuss this range of issues. Too often the venues for managers and leaders to get together and discuss agile issues are tightly focused on delivery and technical topics, or they are so heavily presentation oriented that little connection time remains. At the Agile Executive Forum in August 2011, we found that allocating significant blocks of time for just talking was greatly appreciated. We kept presentations short and interspersed them with lots of time for talking. Just as developers and testers want to connect with peers to discuss issues of importance to them, so leaders and managers want to discuss topics in a peer-to-peer fashion.

I don't claim that these four topics are the total of those considered important by managers and leaders. I don't even claim that they are the most important four. However, they do arise from my experience in a number of different types of venues, in many locations, with a variety of leaders and managers over the last 6 to 8 months.

THE LAW OF RASPBERRY JAM: REFLECTING ON AGILE PROGRESS

In his classic *The Secrets of Consulting* (1986), Jerry Weinberg offers us his Law of Raspberry Jam: "The wider you spread it, the thinner it gets." I've thought about this point recently as I've read blogs and articles from agilists who are bemoaning the state of the agile movement. They are concerned that the movement has gone awry, that people are practicing prescriptive agile, that they are not living up to the vision of the founders.

So what did they expect?

As any movement expands from its narrow early base of practitioners, others take it in unforeseen directions—some good, some not so good. That's just the way movements go. We can wax nostalgic about the "good old days," we can reflect on the progress made and try to redirect the movement, or we can innovate and move forward. As we reflect on 10 years of agility, I'd prefer to focus on the positive—how we've learned to deliver value to customers faster, how we've brought quality to the forefront in ways that haven't happened before, and how we've improved the quality of workplaces around the globe.

Yes, it's thinner than we would like. Of course, thinner isn't all bad, and there are plenty of individual companies and organizations that are thick. Jam (at least good jam) is lumpy—it's thicker in some places than others. Let's push forward on things like DevOps, continuous integration, Lean/Kanban, agile product management, and enterprise agility. Let's stand on the jam lumps and create more of them.

UNORTHODOX, UNCONVENTIONAL, AND THE NEXT DECADE OF AGILE

Paul Farmer was an unconventional doctor. Tracy Kidder, in his classic *Mountains Beyond Mountains* (2009), describes Farmer's Haitian clinic, located in the difficult-to-access highlands, where Farmer often hiked for hours to see a single patient or treated multiple-antibiotic-resistant tuberculosis patients with expensive new drugs. Farmer didn't follow the typical public health cost–benefit approach; he treated individuals and riled that very public health community with his unconventional

approach. In bringing medical services to the desperately poor, he changed health care delivery on three continents.

In *Different: Escaping the Competitive Herd* (2011), Harvard Business School professor Youngme Moon writes about "succeeding in a world where conformity reigns but exceptions rule." Product differentiation, meaning sustainable differentiation, she says, "is rarely a function of well-roundedness; it is typically a function of lopsidedness."

Recently, well-known blogger Seth Godin wrote about unreasonableness examples: "It's unreasonable to start a new company without the reassurance venture money can bring. It's unreasonable to devote years of your life to making a product that most people will never appreciate." But, he notes in his final comment, you have to compete against companies that are unreasonable.

Walk through your neighborhood grocery and look at hand soap—hundreds of variations, little differentiation. Innovation requires stepping out of comfort zones and being different from others. I know one IT organization that has figured out how to capitalize more than 90% of its software development expense by looking beyond tradition. I know another IT organization that has eschewed the common offshoring wisdom, keeping development onshore and outperforming its competitors.

Ten years ago, in February 2001, a group of 17 unconventional, unorthodox, and sometimes really strange individuals got together, wrote the Agile Manifesto, and launched the agile movement. In the last 10 years, agile delivery has often moved from the unconventional to the conventional, from the maverick to the conformist. So what now?

The roots of agility lie in complex adaptive systems and the notion of operating at the edge of chaos, that knife-edged balancing point between chaos and stifling structure. As we move into the second decade (wow!) of the agile movement, we can't forget that agility isn't about structure, practices, and conventions. Instead, agility is ultimately about living on the edge, pushing the envelope, standing out in a crowd, and being lopsided in a world of conformity.

INNOVATORS, IMITATORS, AND IDIOTS

In a PBS interview concerning the financial meltdown of the early 21st century, Warren Buffett commented on the natural progression of how good ideas go wrong. He called this path the "three I's." "First come the innovators, who see opportunities and create genuine value. Then come the imitators, who copy what the innovators have done. Sometimes they improve on the original idea; often they tarnish it. Last come

the idiots, whose avarice undermines the innovations they are trying to exploit" (Taylor, 2011).

Buffett's assessment sparked my thinking about where the agile movement is positioned in this progression after a decade of evolution. We are more familiar with the technology adoption curve—enthusiasts, visionaries, pragmatists, conservatives, and skeptics. Many pundits project that the agile movement has crossed the "chasm" (a term popularized by Jeffery Moore) into wide acceptance in the industry. But what about Buffett's progression? Are we in danger of being overtaken by the imitators and the idiots?

There has surely been a large influx of imitators into the agile movement, an inevitable trend as the market for agile services and tools has expanded rapidly. As Buffett's quote indicates, many of these imitators have added improvements, but some have tarnished the agile brand. Moreover, there have been a few idiots—people and companies that barely know how to spell "agile" hanging out their agile shingles, often giving agile delivery a bad name in the process.

But the real question is, How do we keep moving forward as a movement? At least four key ways come to mind: continue to innovate, balance idealism and practicality, reinvigorate our agile value roots, and unify rather than splinter.

First, I'm encouraged by the innovation that continues to occur: DevOps, continuous delivery, the conversations over technical debt, scaled agile, Lean, Kanban, agile/adaptive leadership, and more. Continued innovation combats the creep of staleness that tends to infect movements after a few years.

Second, particularly as agile permeates into larger organizations, we have to focus on both idealism and practicality. Many people don't care much about esoteric arguments—they care about results. Idealism and innovation are absolutely necessary for a vibrant movement, but they need to be balanced with a dose of practicality. For example, I worked with a large organization that had a corporate-wide phase-gate process for project governance. It was able to work out an agile version for software development that still fit within the overall governance process. The idealistic approach might try to eliminate the phase-gate process, but that would not have been acceptable to management and it would have damaged the credibility of the agile roll-out.

Third, the power and attractiveness of the agile movement lies in its values as expressed in the Agile Manifesto and the Declaration of Interdependence. The more we can emphasize the dual importance of both doing agile (practices) and being agile (values), the better prepared we will be to move forward on a solid foundation.

Finally, as any movement grows, there comes a time when it tends to splinter and a time (sometimes) when it unifies. I appreciated Mike Cohn's recent Scrum Alliance update when he said, "We want Scrum teams to look beyond the Scrum framework and experience the great ideas found in our sister approaches of Lean, Extreme Programming, Kanban, Feature-Driven Development, DSDM, Crystal, Adaptive, and more." Efforts like this to bring the agile/Scrum/Lean/Kanban/other communities together, rather than continue to splinter further, leaves less space for the idiots to exploit.

Where are we in the innovators–imitators–idiots' progression? The answer may not be as important as the question. The trick seems to be to move back and forth between innovators and imitators—advancing and then consolidating—without falling into the idiot trap as did the financial industry. Focusing on agile values and principles, continuing to innovate, balancing between idealism and practicality, and taking opportunities to unify rather than splinter should help keep the idiots at bay.

ITERATIVE DELIVERY, WATERFALL GOVERNANCE

As agile methods become widespread in organizations, the debate over serial, waterfall life cycles versus iterative life cycles is moving from an engineering-level conversation to an executive-level discussion. In terms of project governance, executives are interested in two things—investment and risk. They have to answer two basic questions: "What is the projected value or return on investment (ROI)?" and "What is the probability that this return can be achieved?" Investment decisions are linear: spend some money, receive some results, and decide on the next investment increment. Dollars and time are not iterative: when they are spent, they are gone.

Operational delivery is about defining the best methods of delivering project results. Engineering, whatever the product, is inherently iterative: think a little, try an experiment, observe the results, revise. Sometimes the iterations are long, and sometimes they are short, but engineers have never really operated on a linear, waterfall model—unless forced to do so by an organization process. When development was sequential, the need to differentiate between governance and operations was masked. However, as organizations began to implement iterative methods, the disconnect between governance and operations began to cause friction between executives and project teams. The critical issue for organizations, then, is bridging this seeming gap between linear investment decisions and iterative/agile product development. The solution is separating governance from operations, as shown in Figure 2.2, and then loosely coupling them, thereby abandoning the tight coupling that led to the trouble in the first place.

Product development balances opportunity and risk—the opportunity to generate significant ROI and the probability that something will intervene to undermine the opportunity. That a huge percentage of new product development (NPD) projects fail

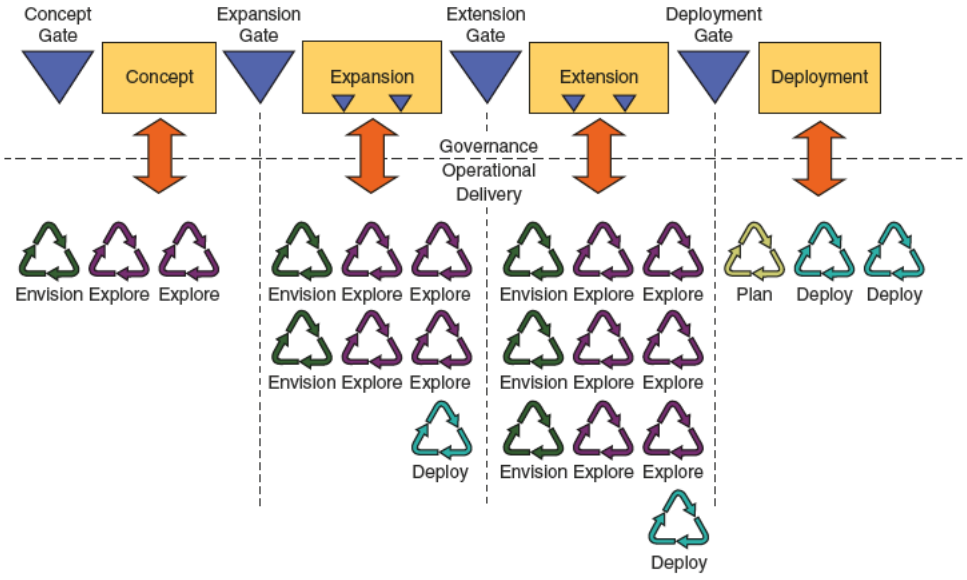

Figure 2.2 Iterative Delivery, Waterfall Governance

speaks to its difficulty. Executives use various data to make project funding decisions. Some known information can be assembled into planning documents, but the crux of product development rests on discovering the unknowns, the solutions to problems that have yet to be identified.

For an uncertain project, executives might authorize a $100,000 expenditure for a concept phase to gather enough information to reduce the risk of making a $5 million full product development decision. Executive decision making follows this scenario through the various phases: define an information gathering strategy based on key risk areas, then decide which activities to fund based on mitigating those risks as early and with as little expenditure as possible. From an executive perspective, this model is a serial one: spend money and time, obtain information, decide on continuing the project.

From an engineering perspective, however, a linear model doesn't serve the purpose well. In the preceding example, a serial model would have focused initially on specifying the requirements for the entire product in detail. At the end of this requirements phase (at which point possibly 20–25% of the cost would be expended), the executives would have relatively complete (in theory) information on the product's requirements, but no information on critical design issues. After expending $1 million, they

wouldn't have reduced the key risks. The team that actually did some design work, constructed a prototype, and tested it might find the key problem was in a component that they hadn't suspected. In a serial project, the team might not have discovered this critical issue until 80% or 90% of the project had been completed, so that it would probably cause significant delays. By developing an iterative prototype, the problem might be uncovered 10% of the way through the project for $100,000 and not cause a project delay.

The funding model for projects should focus on what executives need to carry out their oversight and fiduciary responsibilities. They need a systematic way to view information gathered at key intervals to make the best investment decisions based on their understanding of the risks involved. The delivery model should be the best way of generating that information. In the past, both models have followed a waterfall approach, but a better solution is to use an iterative approach to generate the investment and risk data that executives then use on a regular basis to make project continuation decisions.

IT's Changing Value Proposition

The MIT Center for Information Systems Research issued a very interesting research briefing titled "The IT Unit of the Future" (Ross, 2010). Nearly 90% of respondents in this study indicated that IT's value proposition would change in the next 3 to 5 years. Part of the study, developed from interviews with 50 chief information officers (CIOs), identified four IT value propositions, as shown in Figure 2.3:

- Business support. Utilizing SaaS and the growing technology capabilities of business partners, IT value focuses on coordination and pushing development responsibilities to the business partners.
- Solution/platform delivery: IT value creation is more traditional, coming from building and deploying systems and capabilities.
- Business process design: IT value creation focuses on enterprise process design and optimizations, outsourcing much of the build function to vendor partners.
- Revenue generation: IT value creation focuses on developing digital products and services to enhance revenue generation.

The news from this study was the shift between these four value propositions. As Figure 2.3 indicates, both business support and solutions/platform delivery were expected to account for almost half of the percentage of value creation in 3 to 5 years, whereas the business process design percentage nearly tripled and the revenue generation percentage quadrupled over the same period.

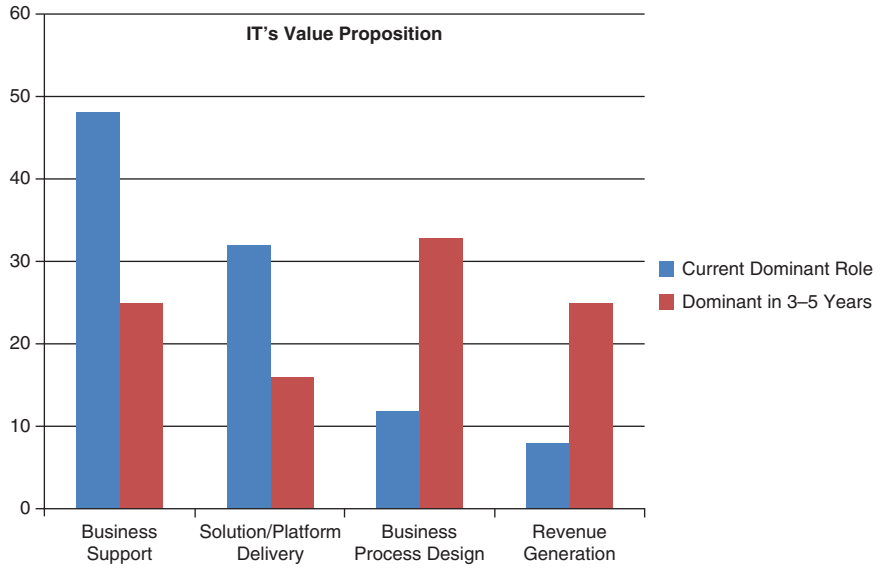

Figure 2.3 IT Value Proposition Changes [Source: Ross, J. C. (2010). The IT unit of the future. Center for Information Systems Research, X(12).]

The implications of these changes in emphasis could be significant in terms of mindset and capabilities in and out of IT departments. From a focus on standardization, optimization, and cost control, the focus shifts to innovative uses of emerging technologies such as social media, cloud computing, and mobile devices; speed to market; flexibility to follow changing opportunities; and building new products and services.

The re-skilling of IT staff—from technologies to business understanding to managing complex projects—will be critical to making the transition to generating IT value in the future.

MORE THAN SOFTWARE: INTEGRATING ECONOMICS, PRODUCT, AND SOCIAL RESPONSIBILITY

We often compartmentalize our lives—work goes here, politics goes there, social responsibility goes somewhere else. This compartmentalization may appear chaotic, but it is seemingly necessary in organizational life. But is it—necessary? A growing number of companies are seeking to help people integrate these multiple facets of their lives, companies such as Ben & Jerry's and ThoughtWorks.

This integration of product, economic, and social components creates a specific culture in organizations, one that contributes to and actually defines organizational

health. It raises the question, "Is organizational health important to success?" "Organizations that focused on performance and health simultaneously were nearly twice as successful as those that focused on health alone, and nearly three times as successful as those that focused on performance alone." This quote, and in fact the entire book from which it is derived, *Beyond Performance: How Great Organizations Build Ultimate Competitive Advantage*, by Scott Keller and Colin Price (McKinsey & Company), builds a case for the importance of organizational health as key to building high-performance results (Keller, 2011).

In *Drive: The Surprising Truth about What Motivates Us*, Daniel Pink (2009) stresses that three essential components in engaging employees are autonomy, mastery, and purpose. Do companies want to make money to further their existence (certainly the historical case), or do they have a purpose, a raison d'être, that making money helps them achieve? Facebook's purpose, for example, is stated as "To make the world more open and connected," not to make money. Wall Street often sees things differently, but in the future battle for customers and talent (the critical dimensions of competitiveness), purpose draws talent and keeps those talented individuals engaged. People want to work for companies that care about more than just making money. Money is a satisfier, not a motivator—at least for a growing number of us.

Ben and Jerry's, for example, has a three-part mission statement[1]:

- Social: To operate the Company in a way that actively recognizes the central role that business plays in society …
- Product: To make, distribute and sell the finest quality … and promoting business practices that respect the Earth and the Environment.
- Economics: To operate the Company on a sustainable financial basis of …

As part of the company's social mission, Ben & Jerry's website voices support of the Occupy Movement.

ThoughtWorks' mission is "nothing less than to better humanity through software," and is founded on Three Pillars (shown in Figure 2.4):

- Run a sustainable business
- Champion software excellence
- Promote social justice

Providing software engineering training for Ugandan students is an example of how ThoughtWorks combines pillars 2 and 3.

1. Ben & Jerry's Homemade, Inc. Reprinted by permission.

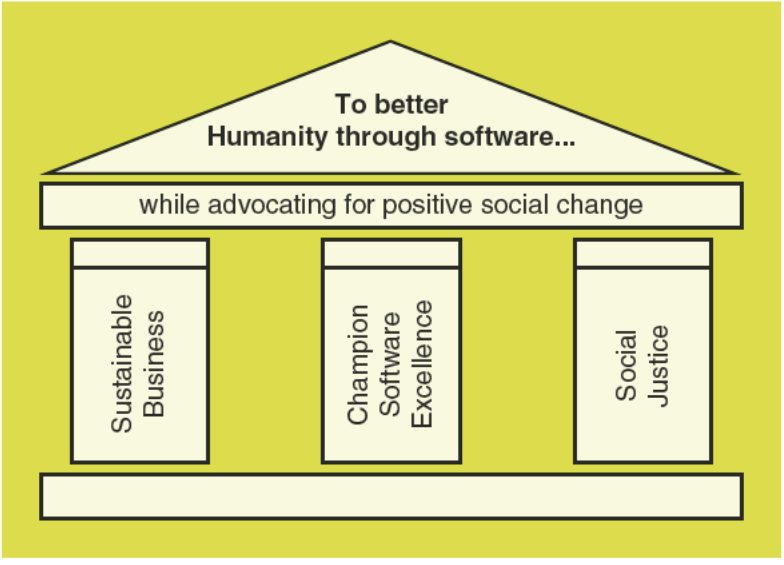

Figure 2.4 ThoughtWorks' Three Pillars

Some individuals won't agree with, or like, Ben & Jerry's stand on issues. Some may not agree with, or like, the organizations or causes that ThoughtWorks supports. But they may find other organizations that take stands they can relate to and agree with. The point is that as companies, organizations, and individuals, we shouldn't have to separate our visions of social and economic justice from our business and professional lives. Integrating the two is good for personal satisfaction, it's good for company performance, and it makes the world a better place for all.

VELOCITY IS KILLING AGILITY

As I have talked with companies around the world, it has become clear that a significant number of them are still mired in the productivity, efficiency, and optimization mud. It's easy to spot these companies because they are often maniacal about measuring velocity—team velocity, velocity across teams, rolling up velocity to an organizational level or even velocity per developer (yuck). Velocity is thereby killing agility. It's the ultimate in applying a reasonable tool for the wrong reasons:

- Velocity is increasingly being used as a productivity measure (not the capacity calibration measure that it was intended to be) that focuses too much attention on the volume of story points delivered.

- Focusing on velocity detracts from the quality of the customer experience delivered and investing enough in the delivery engine (technical quality).
- Giving the product owner/manager complete priority control makes the problem worse—we have gone from customer focus to customer control, which further skews the balance of investing in new features versus the delivery engine.
- Particularly for those parts of the business for which high responsiveness (a deployment cycle time of days or weeks) is critical, investment in the delivery engine is as critical as investing in new features.
- Management needs to allocate resources between features and engine work, and then create a product ownership team consisting of the product owner and technical leader to do feature prioritizing.
- Value (value points) and cycle time metrics will help balance the detrimental effects of velocity measures.

Overemphasis on velocity causes problems because of its wide use as a productivity measure. The proper use of velocity is as a calibration tool, a way to help do capacity-based planning, as Kent Beck describes in *Extreme Programming: Embrace Change* (2000). Productivity measures in general make little sense in knowledge work—but that's fodder for another book. Velocity is also a seductive measure because it's easy to calculate. Even though story points per iteration are calculated on the basis of releasable features, velocity at its core is about effort.

While agile teams try to focus on delivering high-value features, they get sidetracked by reporting on productivity. Time after time, I hear about comments from managers or product owners: "Your velocity fell from 24 last iteration to 21 this time—what's wrong? Let's get it back up, or go even higher." In this scenario, velocity has moved from a useful calibration tool (What is our capacity for the next iteration?) to a performance (productivity) measurement tool. As a consequence, two things get shortchanged: the quality of the customer experience (quantity of features over customer experience) and efforts to improve the delivery engine (technical quality).

Agility is the ability to both create and respond to change so as to prosper in a turbulent business environment. It means we have to build and sustain a continuous value delivery engine, not one that bangs out many features in the first release but then rapidly declines in ability thereafter. The ultimate expression of agility from a software perspective is continuous delivery and deployment. Our goal should not be productivity, but rather "Design and deploy a great customer experience quickly—again and again over time." To respond to business, technology, and competitor turbulence in the market, we have to focus on delivering a superior customer experience, building new features, and improving the delivery engine that allows us to deliver quickly each and every release cycle.

Compounding the problem, the agile movement has focused on high levels of customer involvement—basically a good thing—but we've gone too far. A large number of agilists decry the fact that they can't get organizations to focus on technical practices—but why should that come as a surprise when we encourage product managers/owners to make all the priority decisions and then measure performance using velocity? We have overcompensated for the lack of customer focus in traditional methodologies by giving control of prioritization to the product owner/manager. The tendency has been to lump all work under the heading of "customer-facing" stories and assume we can convince product owners to agree to all the technical engine work that needs to be done. This is an over-correction from traditional projects in which months of technical work preceded any customer-facing work (which was an even worse problem).

Product managers/owners understand prioritizing customer experience work, but they generally have no incentive for undertaking and no understanding of the technical engine work to be done. We need to create a product ownership team consisting of the product owner and a technical leader (which may include a quality assurance lead). Product owners are the customers of today, while technical and quality leaders are the customers of tomorrow.

Imagine our software delivery engine as a high-powered jet engine in a fighter aircraft. What if we fail to perform adequate maintenance on that engine? It gets gunked up. What if we don't periodically rebuild parts of the engine? In software delivery, we gum up the engine by ignoring technical debt, delaying refactorings, disregarding automated testing, underinvesting in continuous integration tools and processes, and accepting long deployment cycles. These things are often considered "technical niceties" rather than keeping the engine running at its optimal capacity.

In Don Reinertsen's book, *The Principles of Product Development Flow* (2009), the author suggests the following formula: Cycle time / Value-added time = 1 / (1 – Utilization). When teams optimize their process for velocity, then, they increase the amount of waste in the delivery process and increase the cycle time. Conversely, when teams optimize for cycle time, utilization (and thus velocity) declines. There has to be a balance.

Business has moved from a world of periodic change to one of continuous change. In parallel to this trend, software development is evolving from project work (deliver software in a big batch and then maintain it) to product-oriented work (deliver software in continuous releases). To drive behavior in the right direction, we need to incorporate measures such as value, delivery cycle time, and technical quality metrics into our performance criteria. We need to calculate feature "value" points as well as feature "effort" points (story points). Cycle time can be an excellent measure because

it depends on the software delivery engine working well, plus it's a measure business customers understand. When we say, "Do we want new features or quality (or technical debt reduction, or something else)?" it's easy for product owners/customers to respond, "Of course, new features!" Instead, we need to say, "How do we need to balance new feature delivery with cycle time reduction?" (Cost of delay, per Don Reinertsen's book, can also be a useful metric.)

This book should not be interpreted as anti-velocity; it still has a place for capacity planning. The problem is the weight given to velocity and the trend of turning it into a productivity measure. Because it is easy to measure, we measure it. But even more important is measuring things such as feature value, feature delivery cycle time, and quality (e.g., defects, technical debt). Delivering high-value features to customers should be paramount, but the focus should not be one-time delivery, but rather continuous delivery. The debate is not features or quality; it's balancing delivering features quickly today with increasing our ability to deliver features quickly tomorrow—over and over again. Business responsiveness is a capability, not a one-time event. Supporting that ongoing need for business responsiveness requires a high-powered, well-oiled software responsiveness engine.[2]

YOU CAN'T PLAN AWAY UNCERTAINTY

Launching my cycling activity in a new area around Boulder, Colorado, I was reminded of an article I wrote several years ago on Hudson Bay Starts. Riding in the Boulder area has infinite possibilities—hundreds of miles of off-road bike trails and a maze of roads with ample to no bike lanes. There are lots of ways to get lost or wind up on a high-traffic, no-shoulder road! My exploration approach is to do short rides out and back, expanding my range with subsequent rides—gradual exploration, if you will. I've spent hours poring over maps and ride descriptions, but getting wheels on the ground is the only way to understand the difference between the map and the territory.

My exploratory rides are similar to Hudson Bay Starts (HBS, but not Harvard Business School) as a way of managing uncertainty. HBS are mentioned in Robert Fulghum's (1991) book, *Uh-Oh: Some Observations from Both Sides of the Refrigerator Door*. They reflect on the nature of the famous Hudson Bay Company that plied the north woods of Canada for nearly a century beginning in the late 1600s. The company had a "willingness to take adventuresome risks and its careful preparation for those risks," writes Fulghum.

---

2. Thanks to Martin Fowler, Jez Humble, and Ken Collier for their comments and ideas on this section.

A Hudson Bay Company expedition would set out to spend months in the wilds of Canada, traveling mostly by canoe. To ensure nothing catastrophic had been forgotten, the adventurers would camp the first night or two within a short distance of their departure point. By setting up camp, cooking, and performing other camp activities, the group had a good chance to figure out if anything major had been forgotten. They could then hurry back and retrieve any forgotten items the next day. Also, because the preparation prior to departure was usually hectic, they would gather that first evening out to discuss and review the purposes for the trip and details of how they planned to carry it out. They were exploring into uncertainty—and their plans. No matter how well they planned, the brief starts usually uncovered things that might have jeopardized the long trips.

Part of Fulghum's book title—"Uh-oh"—also describes a philosophy: "It says to expect the unexpected, and also to expect to be able to deal with it as it happens, most of the time." Sounds a little like adaptive leadership.

This HBS approach speaks to a key concept in agile/adaptive leadership: "You can't plan away uncertainty; you execute away uncertainty." In any situation, there are things that are known, knowable, and unknowable—and it's important to distinguish among the three, at least in concept, because knowing what's knowable and what's unknowable isn't always easy to know. That's why Hudson Bay Starts, iteration 0, and proof-of-concept iterations are so important—they help expose the unknowns and reduce their risk.

For projects, iteration 0 and a couple of development iterations serve the same purpose as a HBS—they force the team to exercise all the activities, all the customer interactions, all the tools. On a project done in a more traditional mode, a single activity—requirements gathering, for example—might take months. By the time the team exercises other aspects of the project—actually building the product, for example—a significant amount of money may have been spent with very little uncertainty reduction. Reducing uncertainty and risk early and at a low cost should be the goal of any business endeavor. Lean Start-ups work in the same way.

HBS hold another appeal for me: they are simple, but powerful. I'm drawn to practices that are simple, or appear simple on the surface. A HBS effort should be relatively simple—pick a couple of uncertainty areas on your project, select a short time frame, and execute a small piece of the project. That part is simple. What it uncovers about your project, your project team, your process, your project objectives, or your organization may not be.

THE AMBIDEXTROUS ORGANIZATION

The agile community has struggled to find a model for moving large organizations to an agile approach to software delivery and face a more daunting struggle in striving toward enterprise agility. Should we strive to convert everyone in a large organization to agile? If not, which parts should be converted and which left to their traditional mode? If a partial conversion is undertaken, will part of the organization feel as if it's second class—not on the cutting edge? If the converted piece of the organization is too small, doesn't it face the organizational antibodies undoing the transformation?

These organizational questions have all been considered as big organizations have tried to "go agile." I recently read several articles about an organizational concept called the ambidextrous organization, described in the *Harvard Business Review* (Tushman, 2004). The author defines two types of initiatives—exploitive ones that focus on the present and exploring ones that focus on the future. I've used the words "efficiency" and "responsiveness" to describe these types of initiatives.

As we know, the cultures for these two types of endeavors are very different, which in turn causes problems in transformational efforts. One historical solution to this problem was to establish a "skunk works," a separate small group that operated outside normal organizational boundaries, processes, and procedures. The problem with these groups is that they were often too small and not supported by other parts of the organization. For example, a "rogue" software development group might use agile methods for a project or two, but not be supported by product management, operations, or the database management group. Similarly, a separate new product organization might not be adequately supported by manufacturing or sales.

As shown in Figure 2.5, the ambidextrous approach focuses on setting up a separate organization—containing all necessary functional units within it and connected

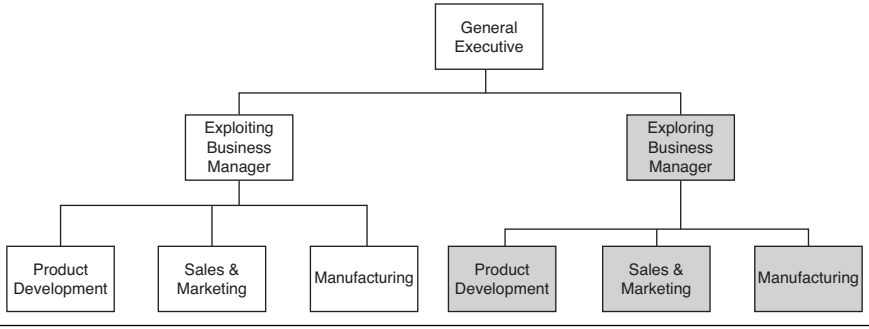

Figure 2.5 Ambidextrous Organizations

to the larger organization at the executive level. In Tushman's (2004) studies, more than 90% of the companies that used an ambidextrous approach met their goals. The exploring organization has enough resources to go to market without help from the exploitive part of the organization; it is a separate business unit that can create different processes and grow a different culture.

This dual organizational structure may have a place in certain agile transformations, particularly in very large organizations that have a need to balance both exploitive and exploring initiatives. It could also be used as an early agile transformation strategy, beyond the more common strategy of just building anywhere from one to several agile teams. The agile team building strategy often runs into problems because it doesn't involve enough other functional groups, such that the agile teams become isolated and discouraged. This approach has some downsides, but it is worth investigating for your agile transformation.

Chapter 3

Deliver a Continuous Flow of Value

In a turbulent and uncertain world, it is no longer adequate to deliver value; an organization must deliver a continuous flow of value, in ever-shortening cycles. Companies must figure out how to take advantage of the continuous flow of opportunities that pass by. Some opportunities linger for years, while others come and go quickly. Adaptive organizations possess the abilities to both sort good opportunities from bad and—this is a critical piece—get the timing right. Organizations need the processes, practices, culture, and leadership to sort through hundreds, if not thousands, of opportunities and turn those potential ventures into specific ambitious missions. These missions are achieved by executing on the right opportunities at the right time. Doing this over and over again, opportunity after opportunity, creates a continuous flow of value for organizations.

Every phase of the business must operate iteratively in short cycles—from the initial identification of new opportunities, to the discovery of product requirements and business models, to the actual product development, to deployment. Companies must hone their product delivery and technology infrastructure capabilities to deliver value time after time after time. No longer will a focus on just the short term or just the long term be adequate. Instead, a simultaneous focus on multiple time horizons, a balancing of short-term solutions with quality, will enable future solutions to be continuously delivered.

The IBM study mentioned in Chapter 1 pointed out that CEOs expect technology, and the changes brought about by technology, to be a critical issue for the next 3 to 5 years. It is somewhat vexing that today many of the opportunities arise from technology, as do many of the solutions to those opportunities. Technology both disrupts and enables. We need to be better at anticipating the former and encouraging the latter. Further, technology today—from mobile devices to implanted medical devices—is driven by software. While the iPhone is a brilliant piece of hardware and its manufacturing process is highly refined, without iOS (the operating system) and applications the iPhone would be just a pretty paperweight. Thus this section on delivering a continuous stream of value focuses on software delivery processes and practices, especially those processes and practices that managers need to understand. In today's world, executives and managers, no matter which department they are in, need to understand the critical software delivery levers.

While the activities of an adaptive leader might seem endless, four critical actions contribute directly to continuous value delivery: improving speed-to-value, having a passion for quality, doing less, and building the necessary capabilities. This chapter addresses these critical actions.

SPEED-TO-VALUE

For our purposes, speed and value both merit further explanation. However, the first order of business is to examine our long-held beliefs about what constitutes performance in both business and projects. If our business objectives are to be responsive and agile, then we should start by examining how we measure success.

Flickr deploys software changes multiple times per day—and advertises this fact on its website. A medical software company deploys versions of its application software more than 75 times per year. Salesforce.com has gained a competitive advantage through its highly automated continuous integration, testing, and deployment of new software features. All of these companies, and a rapidly growing cadre of others, are reaping the benefits of speeding value to their customers.

Achieving speed-to-value reaches far beyond the early agile focus on development speed. In the early 1990s, I worked on a project at a large New York City bank. We managed to build a new application in 6 weeks (using 1-week iterations). The customer was ecstatic, IT operations not so much. After much negotiation, ops agreed to deploy our new application—one the client really wanted quickly—in 6 months.

Speed-to-value embodies two key concepts: speed and value. "Value" means that we are constantly evaluating—at a portfolio, project, capability (epic), feature, and story level—the value we are delivering to customers. It incorporates everything from calculating the ROI of projects to determining the relative (or monetary) value of features. Then on a release, milestone, and iteration level, we constantly prioritize and adjust scope based on value and cost.

The agile mantra has always been to deliver value early and often, but we have not always pushed that point to the limits of actual deployment and customer solutions. The reasons are more organizational than technical (although there have been significant enabling technical advancements).

The organizational issues are both product life cycle and business customer oriented. Although delivery teams have become agile, marketing, product management, or internal business departments have sometimes been reluctant to change their traditional modes of operation. Some product organizations have completely changed their perspective, however—from demanding commitment to features a year in

advance to accepting the flexibility that agile development allows—and have profited handsomely from that responsiveness to customers.

Similarly, at the back end of the life cycle, development and operations may work closely on smooth transitions from development complete to deployment complete. Value is realized only when features are actually deployed, not when they are ready for deployment. Speed-to-value should be measured across the entire life cycle, from placement on a portfolio backlog to actual deployment.

An even more difficult change may be getting business departments to think through how they can use frequent deployments to their advantage and then change their business processes to accommodate them. They will also need to decide how to articulate the benefits of these frequent deployments to their customers. For some business departments, daily deployments of new features may have significant consequences; for others, it may not. Finding the right schedule of deployments for different groups means business departments will need to become more agile themselves.

These organizational agile transformations are often much more difficult than implementation of agile practices and principles in the engineering department, but their benefits can be extraordinary. As enterprises learn to be more adaptive, agile, flexible, or dexterous, the potential for competitive advantage multiplies rapidly.

BEYOND SCOPE, SCHEDULE, AND COST: THE AGILE TRIANGLE

> It's better to have fuzzy numbers for things that are important rather than precise numbers for things that aren't.

Many agile teams are faced with the paradox of being asked by management or customers to be "adaptive, flexible, or agile," while at the same time being asked to "conform to plan," where the "plan" is a traditional Iron Triangle plan based on scope, schedule, and cost. We ask teams to be expansive, to work closely with customers and respond to them, and to seek value—but then we penalize them for being 10% over budget.

The Agile Triangle, shown in Figure 3.1 and introduced in *Agile Project Management* (Highsmith, 2010), addresses the real goals of projects—producing value that delights customers, building in quality that speeds the development process and creates a viable platform for future enhancements, and delivering within constraints (i.e., scope, schedule, and cost). The Agile Triangle alters how we view success.

First, let's look at value. A number of studies have shown that 50% or more of functionality delivered in software is rarely or never used. Even if some of that

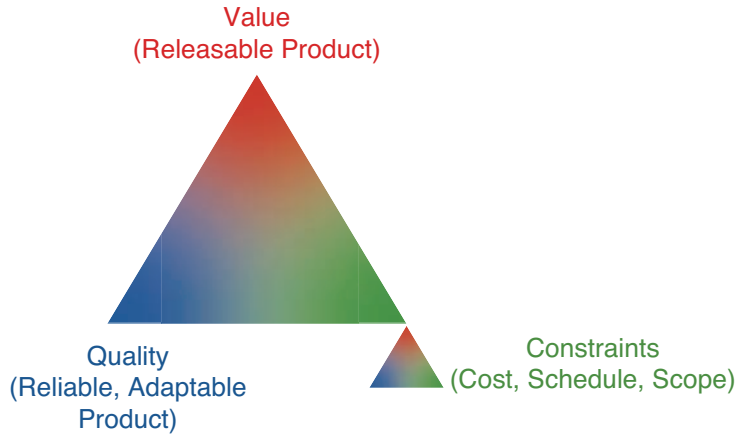

Figure 3.1 The Agile Triangle

functionality is necessary (e.g., the functionality for a year-end accounting close), there is still a huge percentage of unused functionality in most software systems. This leads the conclusion that scope is a very poor project control mechanism—we should be using value. Furthermore, rather than asking, "Did we implement all the requirements?" the question should be "Can we release this product now?" I've known projects that were deemed releasable with just 20–30% of the originally anticipated functionality—and the customers were delighted. They got their fundamental needs met—very fast!

The Agile Triangle also elevates the critical role of quality, a dimension we have given lip service to for far too long. If we are serious about quality, then it deserves a primary place in any measurement program. Quality comes in two flavors—today and tomorrow. "Today" quality addresses the current iteration or release of a product. It measures the reliability of the product: "Does it operate correctly?" If a product operates reliably, it delivers value to the customer in the form of implemented features. Products that are unreliable, ones that give incorrect answers or periodically fail completely, will fail to deliver current value.

The second quality dimension is future quality: "Can the product continue to deliver value in the future?" The ability to deliver in the future tests an application's ability to respond to business changes, both anticipated and unanticipated. While we can often use flexible designs for anticipated changes (e.g., allowing for tax table changes), the strategy to deal with unanticipated changes is different. Responding to the unanticipated future requires adaptability, and the key to adaptability is keeping technical debt low.

The final piece of the Agile Triangle is constraints—scope, schedule, and cost. It's not that these elements are unimportant, but simply they are not the goals of a project. Constraints are critical to the delivery process; they establish clear boundaries within which the team must operate. However, only one of the three elements can be paramount, and on agile projects this is normally schedule.

The Agile Triangle gives us a different way of looking at success, a way that helps resolve the paradox of adaptability versus conformance to plan.

CONSTRAINTS DRIVE INNOVATION

On a recent vacation, I visited the Mingei International Museum in San Diego. During a tour by the museum director, we were looking at a mid-1930s Santo Domingo Pueblo (New Mexico) necklace. "Interesting about this necklace," the director said, "are the 'nots'—not coral, not obsidian, but old melted phonograph records for the black element. During the Depression the Native American artists were constrained by the lack of materials. Everyone thinks that creativity and innovation are driven by freedom. In the art world, they are often driven instead by the constraints."

His comment got me thinking about the constraint point on the Agile Triangle (Figure 3.1). While the most frequent discussions are about value and quality, we can't forget the role of constraints and the way in which they often drive innovation and design. Teams at Ideo, the highly recognized design firm, are often driven by time constraints. While they have great freedom and a highly collaborative environment, they operate under very tight time constraints—and they usually deliver.

There are two key points here:

- Constraints are critical to innovation and creativity.
- Constraints should be carefully constructed.

The second point here is one that often eludes us. Constraints are often arbitrarily set, with little thought to their impact on product development. For example, as Exploration Factors get higher (greater risk and uncertainty), setting aggressive time constraints may be very useful. However, setting aggressive scope requirements at the same time is usually counterproductive.

In the Agile Triangle, schedule, cost, and scope are identified as the main constraints. How we set these constraints has a significant impact on the development effort. Do we set constraints for one of these elements or all of them? Do we set aggressive targets or looser ones? If we want the team to be innovative and creative, setting too many or too aggressive constraints can undermine our goals. One technique

I've found effective is to create a tradeoff matrix that places scope, schedule, and cost along one axis, and fixed, flexible, and accept characteristics along the other axis. Each constraint can then have one and only one value. For example, "fixed scope, flexible schedule, accept cost" would indicate that scope was the most important constraint with limited leeway, schedule would be somewhat flexible (wider tolerances), and cost would be the least important with greater tolerances (but still within acceptable limits). This tradeoff matrix is negotiated between development and product management, and it gives the entire team a good idea about relative importance of the constraints. It helps "steer" innovation and design.

Another type of scope constraint is one that often occurs when replacing an existing system. The requirements are often given as "duplicate the existing functionality." While this may not be the most useful constraint in some circumstances (much functionality may not be used, so it should really be deleted), it is frequently encountered in these projects. Another version of this scope constraint arises when building a product that has direct competition: the constraint may be to duplicate most of the competitor's functionality before releasing the product.

The bottom line is that constraints provide the delivery team with critical information that helps them be innovative and effective. As such, constraints should be carefully considered because they can guide the team to an effective solution—or when done wrong, to one that is awful. Don't forget to think carefully about this third point on the Agile Triangle.

DETERMINING BUSINESS VALUE

The topic of business value is a complex one, and it's easy to get mired in the morass of calculating ROI or in trying to define which intangibles are relevant to your organization. What we need is a model for looking at business value focusing on the portfolio and project levels. Business value is important for a couple of reasons: it helps us focus on what we think is important to our organization, and it helps us make priority decisions among our myriad of opportunities, projects, and product features.

A definition that resonates with my concept of value comes from Wikipedia: "business value is an informal term that includes all forms of value that determine the health and well-being of the firm in the long run. Business value expands concept of value of the firm beyond economic to include other forms of value such as employee value, customer value, supplier value, channel partner value, alliance partner value, managerial value, and societal value. Many of these forms of value are not directly measured in monetary terms."

Ulrich and Smallwood would concur with this last sentence about value measurements. In *How Leaders Build Value* (Smallwood, 2006), this author states that 85% of a company's market capitalization can be attributed to intangible factors such as leadership, culture, and patents. Investors look at the stream of earnings volatility over time to determine price/earnings ratios (which determines market capitalization), and intangibles drive that stream—we just have to look at an intangible like Steve Jobs's leadership to prove the point.

Given the turbulence and uncertainty of today's business environment, picking the right intangible factors can be a daunting task. Nevertheless, one of the key capabilities required is the ability to discover and capitalize on the opportunities that this turbulence creates. A company's ability to take advantage of opportunities requires a number of intangible factors critical to sustaining a flow of earnings.

To summarize, then, business value has both tangible (financial) and intangible components, intangibles are critical to long-term success, and the ability to capitalize on the flow of opportunities is a critical intangible capability for most companies.

To implement this focus on intangibles and opportunities, I propose using a model with Business Value Points (BVP) and a Business Value Points Matrix (BVPM) (Figure 3.2) that would help prioritize projects (e.g., product development). Agilists have long used story points to measure cost. Story points are relative and nonfinancial (sort of). In the past, I've advocated calculating relative value points (Highsmith, 2010) for capabilities (epics) and features (stories are too low level) to indicate that value is important and to help teams prioritize features. But there is an even better reason to use Business Value Points rather than dollars (or euros or yen): it raises the visibility of intangibles and lowers the visibility of financial measures! This is not to

	Start-up	Scale	Mature
Financial	10	40	75
Opportunity	40	20	5
Social	25	20	10
Trait	25	20	10

Figure 3.2 The Business Value Points Matrix

say that financials are unimportant, but rather that other measures are just as important. If Ulrich and Smallwood are correct, focusing on intangibles in the long term has a major impact on financials. Thus we need a model to avoid over-focusing on short-term financials.

Because capitalizing on opportunities is critical to surviving and thriving in our volatile economy, the columns in the BVPM would be "Start-up," "Scale," "Mature," and "Decline." The rows in the matrix would be "Financial," "Opportunity Capture," "Customer Impact," "Employee Impact," "Social Impact," and "Traits," or some combination of these measures customized to your organization (the categories in Figure 3.2 are abbreviated). The numbers indicate the relative importance of the factors. For example, in a Start-up phase, financial results might be relatively unimportant whereas opportunity capture is very important. Conversely, in the Mature phase, financial results might be by far the most important consideration. Most of these factors are self-explanatory, except for traits. Traits are behaviors and capabilities of employees. For example, in a volatile business environment, it might be important for managers and staff to become more adaptable. A project to implement an agile approach to product development would further the objective of increasing the "adaptability" behavior.

This approach changes portfolio management. For example, Mature-phase projects can't be directly compared to Start-up projects—each needs a "bucket" of dollars that is determined by an executive group. Within each category, projects would be prioritized by assigning them a value from 1 to 100, using the table value in each category as a maximum. For example, in the Start-up category, project 1 might have a business value of 55 (financial = 5, opportunity = 30, social = 10, trait = 10), whereas project 2 has a value of 80 (financial = 10, opportunity = 35, social = 20, trait = 15). With a budget for Start-up projects (an entire book could be written on the problem of having a fixed budget for start-ups), a business value priority list might indicate that the "cut-off" for funding would be 12 projects.

While this type of analysis might not be complex enough for financial types, we have to remember the objectives of assigning relative business values:

- Aiding prioritization of projects
- Systematically utilizing both financial and nonfinancial criteria to demonstrate the importance of value

Looking at Figure 3.2, it becomes obvious that a complex, detailed financial analysis would not be useful in evaluating Start-ups, but it would be helpful in evaluating the Mature category. Also, at a portfolio level, each stage of opportunity capture needs a different portfolio management process. For example, whereas projects in the

Start-up phase need fast, frequent review, the Mature-phase review could progress at a more leisurely pace.

A similar model can be used at the capability or feature level, although the criteria may prove very difficult to assess. Traits, for example, while relevant on a project level, wouldn't help us prioritize features. Allocating BVP at a feature level would, therefore, require some adjustments to the criteria matrix.

This section has merely scratched the surface of determining business value. Even so, it offers a model that addresses key issues in today's world—speed, uncertainty, and new technologies, among others—from an opportunity management and intangible focus perspectives.

BEYOND PROJECT PLANS

The agile community has long advocated self-organizing teams. However, the emphasis has been on how teams perform work, make technical decisions, and the like. Most teams continue to operate in the same traditional way when it comes to measuring project performance and the application of controls. If empowerment truly focuses on decentralized decisions and authority, maybe it's time to reevaluate how we empower teams from a financial and performance perspective. In too many cases, we are still binding them to fixed plans, as shown by a traditional Gantt chart in Figure 3.3.

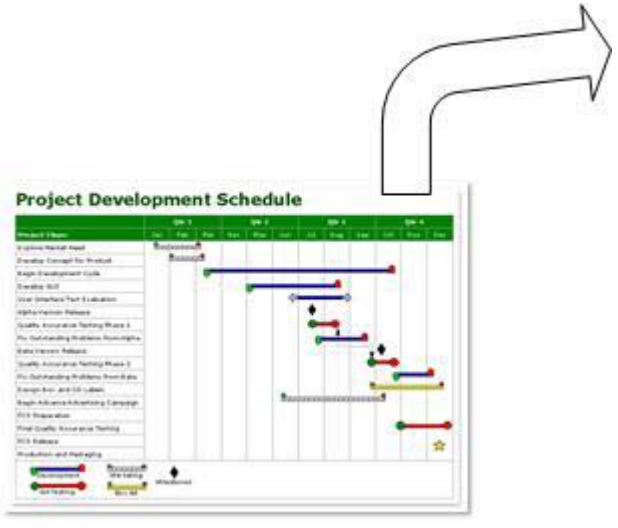

Figure 3.3 Beyond Project Plans

My inspiration for this section comes from meeting Bjarte Bogsnes, Vice President of Performance Management Development at Statoil and author of *Implementing Beyond Budgeting* (Bogsnes, 2008), in Australia at a conference. After talking with Bjarte, I reread his book, which discusses how he helped eliminate budgets in several large companies. One of the insights he gained was thinking about the question, "What do we really use budgets for?" The equivalent question in project terms could be, "What do we use project plans for?" The most obvious answer to that question has three components (you may think of others):

- Coordination with other activities
- Financial and value controls (value is more than money)
- Motivation

The insight that Bjarte and others had was that they were using a single number for multiple purposes and that the single number was causing significant problems. By eliminating budgets, monitoring costs and revenue in new ways, and creating a new set of relative performance guides, companies can break loose from the budgeting straitjacket and improve performance. Maybe there is a parallel for projects.

Traditionally, managers look at three project measures: schedule, cost, and scope. Furthermore, they insist on meeting all three planned measures exactly—a virtual impossibility in today's turbulent business environment. What if we look at three measures for each of these?

- Targets: Desired business outcomes (usually a stretch goal)
- Forecasts: Best current estimate of outcomes
- Constraints: The limits of the team's authority

Let's apply these measures to a project whose traditional cost "budget" is $250,000. We could have a target of $200,000 (might happen if everything goes right), a forecast of $240,000 (our current estimate of the total cost), and a constraint of $275,000 (the team is authorized to spend up to this amount). If the team delivers the project with the capabilities agreed to (value, not scope), with the appropriate quality of results, within the time "constraint," then any cost between $200,000 and $275,000 might be considered acceptable. I say "might" because only a holistic evaluation of the outcomes can determine performance. Give project teams more leeway with results (a lot more leeway), and performance usually improves.

One key determinant of project success is team motivation, and unfortunately most traditional project controls and measures try hard to "demotivate" teams. Consequently, just using the wider limits on the traditional measures cited previously won't be enough. Traditional measures tend to be of the stick kind ("Do this or else"). Time

and again, studies, highlighted by Dan Pink's (2009) work, show that the best motivators are intrinsic—purpose, mastery, and autonomy—and focus on positives rather than negatives. Better motivators for projects are purpose-driven outcomes such as customer value delivery and quality. Better motivators are relative comparison measures, not absolute numbers. Better motivators are those that allow the team autonomy, and that includes leeway on things such as cost, scope, and schedule. Give teams vision, facilitate their self-organization, and provide constraints (loose boundaries), and you encourage them to be creative and innovative in their solutions.

Giving people stretch goals may motivate them, but when those goals are linked to potential punitive actions if not met, their motivational value is lost. Motivation needs to focus on vision and purpose, not penalizing people. When people think they will be judged against plans, they are likely to fudge the plans, or even sometimes the actual performance. Instead, break plans into two numbers, each with a specific purpose: a target based on business needs that represents a stretch goal, and forecasts (updated regularly) that are the best estimates of outcomes.

But, someone always says, what about cost control? (It always seems to be about cost control, not value delivery.) The answer for this is (1) to track actual costs carefully and watch trends, and (2) to establish an authority constraint (limit) for each project. This constraint should be generous, not onerous. Projects that are forecasted to exceed their constraints would need further review and possibly additional funding (or termination). The big difference here is the difference between a predicted cost and a cost constraint.

Project teams also need to coordinate with others (e.g., teams, projects, departments), usually about schedules. Forecasts are the numbers used for coordination. If targets are really stretch goals—with, say, a 50/50 chance of achieving them—then project teams shouldn't base their plans on these targets, but rather on forecasts, meaning our best estimates of outcomes. By using these two measures in tandem, both coordination and motivation are improved.

For this evaluation system to work, two things have to happen. First, everyone, including both managers and team members, has to understand the rationale behind each type of number. Second, everyone needs to evaluate the results holistically. Every number has a purpose, but performance is evaluated by taking all the numbers into account in a holistic manner. No single number, or even a series of numbers, is adequate to completely evaluate a complex undertaking such as a project. Performance evaluation should focus primarily on dimensions such as value and quality, and secondarily on constraints such as cost or scope.

Any metrics system can be gamed. The success or failure of any metrics system lies in the intent of the managers and leaders applying the system. As Rob Austin, dean of

the business school at the University of New Brunswick, says, "If there is a single message that comes from this book, it is that trust, honesty, and good intentions are more efficient in many social contexts than verification, guile, and self-interest" (Austin, 1996).

FEATURE FOLLY

In my executive presentations, I spend a fair amount of time on the topic of "Do Less," talking about how we waste an incredible amount of time and money building features that are rarely or never used. Published studies put this number at far greater than 50%. I can see audience members agreeing with my message, but I can also see the little thought bubble floating slightly above their heads: "but not in MY organization." Organizations worry about improving development productivity by 10% when what they should be worried about is improving customer demand effectiveness by 25%. I've always said that one of the biggest potential productivity improvements from the agile approach lies in all the features that we don't do because of the constant attention to simplicity and delivering the highest value to the customer.

In most organizations, software development efficiencies are subjected to infinite analysis, while customer demand effectiveness isn't mentioned at all. We neither measure nor calculate feature value in the beginning, nor do we measure whether value was actually captured as the customer or product manager had predicted. In many large companies, project ROI might be calculated as part of the portfolio management process, but rarely does anyone follow up to see if that ROI was, in fact, attained. Consequently, development teams have multiple feedback mechanisms, whereas customer/product teams have relatively few, except possibly at a macro level (product sales, for example). Product managers are left with gut feel as the basis for most of their feature-level decisions. No wonder 50% or more of developed features are rarely or never used. Product management teams aren't the culprit here, but rather the lack of adequate feedback mechanisms.

If we look at a typical agile project team, we will likely see a development subteam and a product/customer subteam. Roles, responsibilities, and feedback mechanisms have been defined. For example, the product team identifies stories, writes stories, prioritizes them, and develops acceptance criteria (both automated acceptance tests and feature showcase evaluations). In turn, there are micro-level (feature and story) feedback mechanisms to steer the development effort. But what about feedback from product management to the business? These links are often much more tenuous, such as measures of overall sales that tell how a product is doing at a macro level, but say nothing about individual features. Internal IT products generate even less feedback in most cases. Lack of feedback leads to feature bloat, because it's always easy to succumb to customer requests and internal demands to improve the product.

At the Agile Brazil 2011 conference, I was listening to Josh Keriesvky's talk on his Lean Start-up experience, and gravitated to thinking about this problem. Here's a starter list of ideas about potential solutions:

- Develop customer demand effectiveness measures for every product management organization and team.
- Calculate relative or monetary value for every feature.
- Use relative benefits such as increasing customer happiness, reducing customer risk, and improving the internal collaboration culture to evaluate feature value.
- Build feature usage information into software to provide feedback to product management. Product manages could thereby gain insight into what was actually used and what wasn't.
- Develop feature evaluation experiments into the feature identification and prioritization process. For example, do A/B testing on features.
- Do false feature analysis. Give users access to a feature that, when selected, pops up a message such as "This feature is under development. When it is completed, are you 'very likely,' 'somewhat likely,' or 'not likely' to use it?" Josh talked about an expensive feature that his company decided not to implement because of the results from this type of test.
- Use short, focused surveys to measure customer happiness. For example, ask your customers, "What was your experience using capability Y (this capability being several features): Awesome, OK, Not So Hot."

From a Lean perspective, the biggest "waste" factor in software development is low- or no-value features. If we were able to eliminate 15%, 20%, or 30% of the scope of software projects, we would increase the chances of meeting planned delivery dates. Agile's biggest contribution to productivity, then, might be all the functionality we don't produce.

Of course, agility is not just about cutting out stuff, but about emphasizing the right thing—and that thing is value. Product managers should be calculating business value, either relative or monetary, to the feature level so that teams can produce the highest-value feature early. Then, if scope reductions need to happen later in a project, the eliminated features come from the lower-value list.

While the agile and Lean communities have often alluded to value-based development, the actual practices to support this objective are not yet sufficient. There aren't sufficient feedback mechanisms at the feature level to help mitigate the constant push toward feature bloat. Maybe taking a look at some of the ideas from Lean Start-up and other sources can help.

FEATURES OR QUALITY? SELLING SOFTWARE EXCELLENCE TO BUSINESS PARTNERS

As Figure 3.4 asks, should we deliver features or quality? It's always been difficult getting business partners (from executives to product owners) interested in quality—be that code quality, design quality, automated testing, or technical debt. Software technical excellence numbers (ah, if we just had good numbers) don't mean much to business partners.

Recently I've been adding to the Agile Triangle (value, quality, constraints) the idea that while business partners are only mildly interested in quality or software excellence (such concepts are too esoteric), they *are* interested in cycle time (getting stuff out faster). Furthermore, I hypothesize that cycle time is a function of quality (among other factors). Thus we need to "sell" software excellence on two key bases—valued delivered and cycle time. If cycle time is, in fact, a function of quality, then we should be touting cycle time (the *what*) improvements and leaving code quality, design quality, and technical debt discussions (the *how*) mostly to the engineers.

From a business partner's perspective, the question "Features or quality?" is easy to answer—more features. The partner is being asked to trade off a business outcome (features) for a technical outcome (quality)—something that is easy for the partner to understand versus something that is difficult. It's not a hard choice, as you might imagine. However, the question "Features or cycle time?" isn't so easy to address—because it requires trading off two business outcomes. If we can show the relationship between cycle time and quality and then start measuring and reporting cycle time, perhaps we can give our business partners a better and more realistic way of assessing software delivery performance.

When I was presenting this hypothesis to a group recently, one of my colleagues offered this challenge: "How can cycle time be a function of quality when everyone knows that quality can be traded off for additional functionality—people do it all the time." This challenge stuck with me for several months without a good answer until I

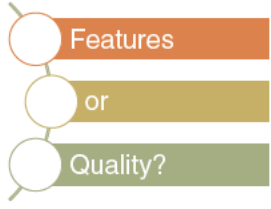

Figure 3.4 Features or Quality

heard Martin Fowler's talk on technical debt. I was mulling all of these issues over on a bike ride when the solution occurred to me.

When people trade off more features for less quality, it's usually for a single release occurrence, not for an aggregation of releases over time. This is an outgrowth of waterfall development, in which intervals between releases were long, often a year or more, and trading new features for lower quality (say, poor design or less testing) obscured the cost and pushed consequences far into the future. When we have a large batch size (hundreds of features) and a long time frame (a year or more), the next releases (small maintenance or enhancements) are so trivial in relation to the first release that the feedback or impact of low quality is very difficult to determine. In a waterfall project, it is easier to cut refactoring, for example, because the impact is felt in the future, and even then only the engineers feel the pain. Cycle time measures are irrelevant when release cycles are too long.

However, as agile teams reduce delivery cycles to months, weeks, and days, the impact of poor quality becomes much easier to determine. When a team is running 1-week deployment cycles, the effects of poor testing in one cycle may pop up quickly, in the next cycle or two. Poor design in one cycle begins to retard feature delivery in the next few cycles—the consequential feedback comes in a few weeks. If a team is measuring both feature throughput and cycle time, either or both can suffer quickly from software mediocrity. However, even in our agile era, not enough teams systematically measure cycle time (other than those who are doing Kanban), so the relationship between quality and cycle time remains murky for many.

If the delivery teams are keeping reasonable metrics, the quality/cycle time relationship becomes clear quickly. What also becomes clear is that the old assumption about features and quality are wrong—that technical excellence can increase throughput *and* reduce cycle time. Waterfall projects cover up this understanding.

Finally, just a brief note to admit that cycle time measures can be thorny. There are three types of cycle time, all important. The first is feature delivery time (from inception to release). The second is release frequency (how often we release and/or deploy the product). Finding the starting and ending points for feature cycle time can be tricky. But these difficulties can be overcome as we learn better ways of measuring. Once these cycles begin decreasing from months to weeks and days, the impact of technical excellence becomes much clearer. The third type of cycle time is product delivery time—from inception to final (or release) delivery. Studies have shown, for example, that business customers of IT have about a 6-month window that they consider "good" performance. Correspondingly, projects with greater than 6-month delivery cycles are usually in trouble from the start—regardless of other factors.

Let's help our business partners move from thinking about "business" features versus "technical" quality to a more productive view of "business" features versus "business" cycle time.

ALL PROJECTS ARE NOT THE SAME

One of the big problems with successfully executing projects is that while we know projects are very different from each other, we often manage them and measure their success in the same way. Think for a moment about the oft-quoted Standish reports in which project success is measured on the traditional Iron Triangle basis of meeting scope, schedule, and cost plans. In the Standish scheme, all projects, of any type, are successful or not based on the same criteria.

A friend of mine worked on a project recently where the client said, "I have a fuzzy vision of what we want. I don't have any idea what the detailed requirements should be. I need results fast." The client even refused to participate in a story identification process, leaving experimentation up to the development team. Managing this project against a backdrop consisting of traditional measures of scope, schedule, and cost would eliminate any chance of success; it is a different kind of project. The team evolved the product from a vision and evolutionary learning and the client was very pleased with the results, because he understood it was a different type of project.

In *Agile Project Management* (Highsmith, 2010), I wrote about assigning an Exploration Factor (EF) to each project (or release of a product). The EF attempts to identify a level of uncertainty and risk for a project by looking at both technology and requirements. Technology can run the gamut from "well known" to "bleeding edge," and requirements can range from "stable" to "erratic." Combining the two factors yields EFs (see Figure 3.5) ranging from 1 (very low uncertainty) to 10 (very high

	Product Technology Dimension			
Product Requirements Dimension	Bleeding Edge	Leading Edge	Familiar	Well Known
Erratic	10	8	7	7
Fluctuating	8	7	6	5
Routine	7	6	4	3
Stable	7	5	3	1

Figure 3.5 Exploration Factors

uncertainty and risk). EF 1 and EF 10 projects are very different, so they need to be managed differently and assessed differently. For example, if the requirements are uncertain, measuring progress against a scope plan is ridiculous. This doesn't mean the project is uncontrollable, just that we have to control it differently.

Before we think about how to measure success on projects with high EFs, we need to understand what makes these projects successful. The pressures on high-EF projects are usually uncertainty (about either requirements or technology, or both) and speed. Typically high-EF projects are also strategic, customer facing, Web or mobile, or some other flavor that tend to increase the uncertainty. Sponsors want them done quickly. How are projects like this managed successfully? By using a combination of a well-articulated vision, quick iterations, functionality evolved to a minimal viable product and beyond, adaptability as a result of learning from customers as the product evolves, focus on value delivery, and time boxing.

One important aspect of controlling high-EF projects is to view scope, schedule, and cost as constraints, not objectives. Because the project requirements are not known, it is often difficult to estimate time. Given the lack of specificity, however, the team needs constraints—for example, a time box of 3 months that limits exposure. At the end of this time frame, the project sponsor can evaluate the value delivered and the questions answered, and then decide whether to invest in the next increment. The question to be asked at every iteration and release is, "What is keeping us from deploying this product now?" This is very different from the normal progress question of "Have we developed all the planned scope yet?"

Conversely, a very low-EF project (say, a 1 or 2) could be managed with scope questions because the requirements are knowable in the beginning (or at least people think they are, even though they are often wrong). Note that software development projects are rarely low-EF projects.

The bottom line is this:

- All projects are not the same.
- Projects with different Exploration Factors should be managed differently.
- Performance measures should be different for high-EF projects than for low-EF ones.

SCOPE ISSUES IN AN AGILE PROJECT

I was talking with a colleague the other day about troubles with scope management in an agile project. She was lamenting problems that were arising with a particular client who was concerned about the progress of the delivery team. Because agile teams

use time-boxed iterations and then let scope adjust to those iterations, this shouldn't be a problem—should it?

A number of issues surround scope management in an agile project, many of which are the same as the issues that arise when trying to manage scope in a traditional project:

- Perception versus reality
- Poor definition of how to measure scope
- Running an agile project in a nonagile organization
- Wish-based planning

Agile projects, unfortunately, don't eliminate the perception-versus-reality problem. Miscommunication and misunderstanding can impact an agile project even as teams try to expose reality—or at least their version of reality. Short iterations and working software can reduce the perception/reality gap, but as long as projects are delivered by people and evaluated by other people, a gap will often remain. Good project managers realize they have to manage both—reality and perception.

Another question that isn't asked or answered very often is, "What is scope?" Is it the number of requirements, stories, or features? Is it the total work hours or story points estimated for the project? Is it the documented requirements? At which point is scope determined, given that in an agile project features can change over the life of the project? These are all bottom-up measures of scope (and therefore progress). Maybe a better approach, especially in an agile project in which the detail stories and features are changing, is to ask a top-down question, "Can we deploy this product at the end of this iteration?" Obviously, the answer to this question involves some determination about feature completion, but the question really asks about value, about enough value to deploy, not about whether a set of detailed requirements has been met.

Scope issues often crop up when agile teams confront traditional organizational success measurements. The teams may view themselves as being successful on the project even as managers are wondering what's going on because they don't understand this "iterative" approach. This is somewhat different from the perception-versus-reality issue; it's more the clash of two different perceptions (or two realities).

Finally, too many organizations subscribe to what I've called "wish-based planning." They do a lousy job of capacity planning—that is, balancing the demands for work to be done with the actual capacity of the organization. These managers don't understand the difference between stretching limits and being completely unreasonable. All too often, then, stretched project plans become irrational wish-based plans. Agile teams will still experience scope problems in this type of dysfunctional situation.

Adopting an agile approach won't fix all your scope problems. Agile development can help teams and management look at scope from a different perspective, but the long-held perceptions of scope will be difficult to change in many organizations.

SPEED

Both speed and value are important. Delivering value early and often (every iteration or set of iterations) can improve ROI substantially over delivering value at the end of a 12- to 18-month serial project. Nevertheless, speed needs to incorporate not just engineering, meaning creating features that are ready for deployment, but also the needs of the business organizations that actually deploy and use the features. Speed might measure the time between order input and order fulfillment, or the time between release of a feature into development and its deployment. The former is a measure of business results speed; the latter is a measure of software features that support this business process.

Speed measurements in Kanban projects—time from release off a backlog into development until the feature is complete (tested, accepted)—are being used in service-level agreements. Because of the strict work-in-process limits, agreements such as "We agree to deliver new features within 21 days with a 95% confidence limit" can be made.

Speed is also about perception, and the elapsed time of a project can be more about perception than reality. When people declare, "The project is late," they may actually mean the project is taking too long, irrespective of the planned schedule. The potential for such a negative perspective to emerge grows as projects lengthen. For example, even though a project is planned for 2 years and is on schedule, the perception of its progress is often negative just because of the overall length of time (of course, 2-year plans are almost never accurate). By comparison, a project that delivers results in 3 to 6 months will usually be well perceived—even if it is a month "over budget." To some extent, regardless of plans, results in a short period are considered successful while projects that roll out over longer periods are considered not successful. Reducing project time frames can, by itself, improve the perception of success—at least in terms of delivering greater speed.

REDUCING CYCLE TIME

An increasing number of organizations are moving toward radical reductions in cycle time as they move toward rapid business responsiveness and continuous delivery.

One mantra that seems to help teams and organizations in this quest is, "If it's hard to do, do it more often!" Keep this mantra in mind. If something appears too hard, too costly, or too slow, figure out a way to do it more often. I once worked with a

company whose product took 6 months of quality assurance (QA) prior to release. The QA manager couldn't imagine how to reduce the time to 2-week iterations, so I asked him if he could figure out how to do it every 2 months. After several subsequent iterations, his group was able to support 2-week iterations. From Lean manufacturing examples, we often see that 80% or more of the time taken to accomplish a process usually works out to be wait time, not work time, so pushing for significant reductions is often much easier than anticipated at first.

In trying to answer the question, "How can we do something frequently?" the answer may come from a combination of simplification, elimination of constraints, and automation. Every time we push to do something more often, whether it is a software build and integration or a design, we learn. Learning comes from repetition.

From Goldratt's (1984) theory of constraints, we have learned to look for process bottlenecks. The bottleneck could be the lack of a particular skill or the throughput of a machine or a computer, but the key aspect of a bottleneck is that its elimination can create significant improvements in overall throughput and cycle time reduction. Conversely, if we don't think about bottlenecks, we can add significant resources without impacting throughput at all.

Teams should always ask the question, "How could we simplify this process or activity?" In some cases, this question may be more like "What can we eliminate or streamline to reduce the time to do this from 6 weeks to 1 week?" Again, the time to accomplish some overall process can often be traced to delays between groups and excessive control processes or steps. For example, if a sequential process takes 8 weeks and involves four groups, each group may have a logging, prioritization, and reviewing process for work items. Reducing the process to 1 week by eliminating communication delays may also eliminate the need for these control processes.

Because we are in the business of IT, automation always comes to mind as an enabler to doing things more often. Despite its allure, we shouldn't jump to implement automation until some of these other ideas have been tried.

The mantra "If it's hard to do, do it more often!" espouses the agile value of responding to change. In today's high-change world, responsiveness to change is tied to the cost of change—reducing its cost increases our responsiveness. The high cost of some changes should not be viewed as a barrier to responsiveness, but rather as an opportunity to increase our responsiveness by overcoming that barrier.

CYCLES, CYCLES, CYCLES

One of the problems in integrating agile delivery or continuous delivery into enterprises is the differences in cycles. In the past, companies have tended to run on annual

budgeting cycles and even longer strategic planning cycles, with some (though not always close) coordination between the two. Product management tends to run on product cycles, which, depending on the product type, can last for months to years (for airplanes, for example). Projects tend to run in phases (traditional projects) or releases and iterations (agile). Project management offices, reflecting upper management's desires, operate on monthly, quarterly, and annual cycles.

There are several potential problems with all these cycles:

- The cycles don't coordinate well.
- Everyone wants everyone else to conform to their cycles.
- Finance considerations, particularly for public companies, seem to drive everyone else's cycles.
- Different kinds of projects and business initiatives need different cycles.

Project cycles have always clashed with financial cycles, as projects just don't naturally finish in December. The cost side of projects has traditionally been reported on a calendar basis, but the value side has often not started until the project has completed. Incompatible cycles have plagued developers and accountants alike—for example, biweekly payrolls don't match up nicely with monthly financial reporting.

Most cycle mismatches are time related, but other types of discrepancies can occur. For example, agile cycles deliver partial results in 2-week iterations. By "partial results," I mean working software, but only part of the application; architectural pieces, but not an entire architecture; or requirements, but only details for the stories done during the iteration. However, many company governance cycles are built around completed results—a complete requirements document, a complete test plan, or a complete database design. Governance systems based on this model make agile projects difficult to "govern" in these organizations because of this cycle mismatch.

In my prior life as a waterfall methodologist (yes, I admit it) in the 1980s, I used something called the Warnier-Orr methodology, which included a bracket-style diagram called the Warnier-Orr diagram. Part of the methodology for resolving a cycle conflict was to determine the "lowest common denominator." For example, in a payroll system that needed to generate biweekly paychecks and monthly financial statements, the lowest common denominator would be days. While this is a simple example, it's amazing how many systems kept data at the wrong level of detail and couldn't generate all the required information for different cycles.

Many agile organizations are using the three-tier model for development shown in Figure 3.6: Iteration (2 weeks), Release (3–6 months), and VisionMap (RoadMap) (6–18 months), with corresponding differences in the granularity of the deliverables

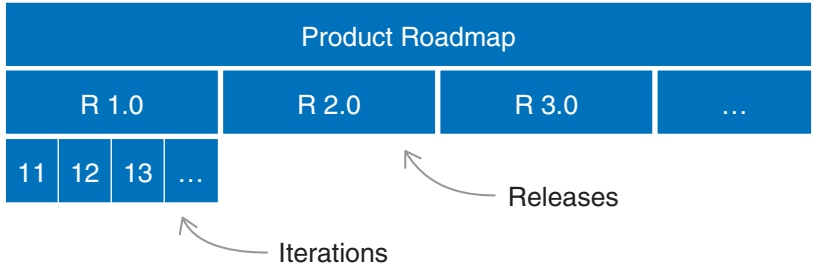

Figure 3.6 Cycles with Different Time Horizons

(stories, features, capabilities). What if the entire enterprise used this "short-horizon" model to actually run the business? What if everyone synchronized their efforts based on 2-week iterations? What if everyone focused on 2-week delivery of partially completed products, documents, business initiatives, plans, and other business artifacts? I was talking with a client recently who was having cycle meshing problems—the company's product management group wasn't syncing up very well with its agile delivery teams. In other companies, DevOps initiatives are attempting to match operations and release management more closely to delivery team releases.

When companies get serious about enterprise agility, one area that will require major change is moving from financial cycles being the driver to focus on a product-driven (deliverables) "short-horizon" model (which delivers the required financial information but is not driven by those financial cycles) that is more adaptable to changing conditions.

SHORTENING THE TAIL

In *Agile Project Management* (Highsmith, 2010), I wrote a short section on a performance metric called "shortening the tail." I liked using the metric of "tail length" because it is easy to calculate and tells a lot about an organization's agile implementation. It's not a vanity metric, like the number of developers who have attended a refactoring seminar, but a true learning metric because it focuses on a key tenet of agile development—running, tested software. It's also a metric that can help an organization move closer to continuous delivery.

The tail is the time period from "code slush" (true code freezes are rare) or "feature freeze" to actual deployment. This is the time period when companies do some or all of the following: beta testing, regression testing, product integration, integration testing, documentation, defect fixing. The worst "tail" I've encountered was 18

months—18 months from feature freeze to product release, and most of that time was spent in QA. I've routinely encountered software companies whose tail is 4 to 6 months of a 12-month release cycle. Other companies, including a growing number of software companies, have honed their processes to have a zero tail length—they are truly doing continuous delivery and continuous deployment. Using the tail length metric, particularly in products or applications that have large legacy code bases, can help organizations monitor their progress toward continuous delivery.

I worked with an organization several years ago that had a 6-month tail in a 12-month release cycle—and the tail was getting progressively worse with every release. The tail had expanded from 4 to 6 months over the past three release cycles. The company set a goal to achieve a 1-month tail and worked diligently to achieve that goal over time.

Shortening the tail is a simple, powerful metric for measuring progress toward agility. The goal of agile teams is to produce shippable software every iteration, but most are far from this goal—especially if they have large, old legacy code bases. Think of everything a company might have to do to reduce a tail from 6 to 3 months, then to 1 month, then to 1 day. The company would have to learn how to do continuous integration across its entire product. It would have to improve its level of automated testing to drive regression and integration testing back into every iteration. It would have to improve the level of automated unit testing done by developers to reduce testing time at the end of iterations and releases. It would have to bring customers into the development process much earlier, rather than waiting until the end for beta testing. It would have to integrate documentation specialists into the team and produce documentation continuously during iterations. It would have to invest in systematic refactoring to reduce the technical debt, and thereby reduce testing and defect fixing time.

You can probably think of more the company would have to do.

Each of these items would contribute in some way, large or small, to reducing the tail by days or weeks, as Figure 3.7 shows. For large products, the tail might never reach zero, but it could be small. Just think of the competitive disadvantage a company has when its delivery tail is 18 months, or even 6 months. Such a lengthy tail means that for 6 or 18 months prior to release, no changes in the competitive environment could be incorporated into the company's products.

If continuous delivery seems too big a step, start on that path by first reducing the tail length on your product releases. Before long, continuous delivery won't seem that big a stretch.

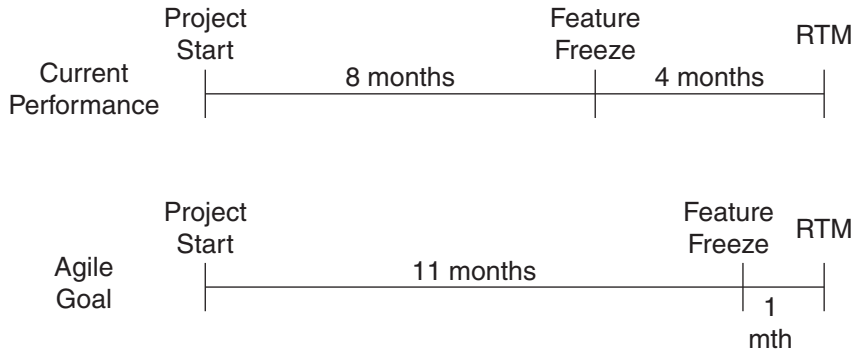

Figure 3.7 Shortening the Tail

QUALITY MANAGEMENT

The more I visit companies and see mangled agile implementations, the more I become convinced that quality, or lack thereof, remains the central issue in achieving effective agility. Organizations begin agile implementations with higher quality as the goal, but then all too frequently they do not carry through with the discipline to achieve the goals they have established. They may have goals and desires, but not enough engagement and commitment. Admittedly, a critical problem is often the relentless pressure on delivery, but managers must step up and begin to transition from the vicious cycle, depicted in Figure 3.8, of "incur technical debt, slow delivery, increase pressure, fail to repay debt, incur more technical debt" to a virtuous cycle of "build high quality, speed delivery, decrease pressure, and repay debt."

Some might ask if management really has an impact on quality. After all, the perception is that quality is up to developers and testers. This next story illustrates that management can, indeed, have a dramatic impact. A large project team that was regularly measuring code toxicity (a combination of several measures) noticed a spike in that toxicity. Tracing back to the time the increased toxicity started, the team members determined that the cause was a change in managers.

When managers talk about quality being a priority, but then fail to allocate money to acquire adequate testing expertise and tools, or fail to emphasize quality with their teams, or fail to allocate time to create and maintain test suites, their lack of real commitment to quality begins to show. When you are caught in the bowels of a vicious cycle, turning that situation around is a management issue. Of course, the technical teams must embrace the requisite practices and discipline, but without managers and executives who are engaged in seeing that quality is critical to the turnaround, teams will have a very difficult time delivering quality products. The strategy needs

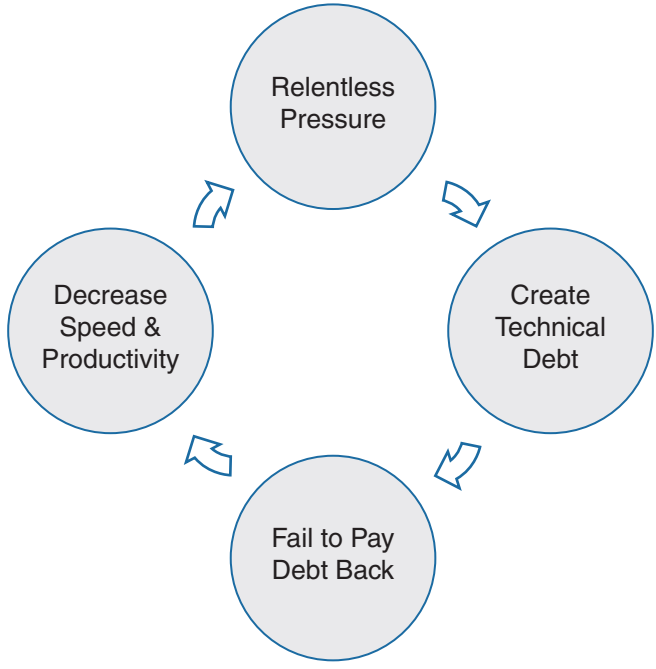

Figure 3.8 Technical Debt's Vicious Cycle

to move from "more features, more features" to "fewer features, higher quality, then more features."

But what does engagement in quality mean? Probably the most difficult task is for managers to really commit to the short-term pain required to deliver long-term gain. The gain may take only months to achieve, but there is always the pain of short-term performance loss (and investment) while people learn new practices. Unfortunately, this pain often leads to lack of full commitment, where managers fail to push their teams to full agile implementation—they get the pain, without the gain. This lack of commitment comes from lack of real understanding of how quality impacts speed, how technical practices fit together (e.g., refactoring, test first, and simple design), and which investment tradeoffs are appropriate. For example, many managers succumb to the pressure to deliver features over quality because they think quality shortcuts are detrimental in the long term, while feature delivery is a short-term issue (short-term gain for long-term pain). From our experience with effective agile teams, we now recognize that inattention to quality begins to degrade delivery velocity in only a few iterations. The road to fast, productive software development goes through quality—a lesson highlighted again and again by metrics gurus such as Capers Jones and Michael Mah, but one that is still not embraced by many practitioners.

Managers are often caught in a perceived dilemma between perfection and "nice to have." On one side, they are often skeptical of what they perceive as the technical team's desire for perfection. They can't discern the difference between perfection and excellence, so they fail to support adequate quality measures. They know whether a feature gets delivered to the customer, but they don't really know how to assess its quality. On the other side, they need to understand the difference between the two aspects of quality—reliability and adaptability—and how to achieve both.

Quality software requires engagement and execution. Execution is the realm of the technical team; engagement is the management side. Managers must do more than say, "Quality is job 1," every 2 months. They must understand what the right quality framework is; they must appreciate the consequences of poor software for customers; they must find the appropriate balance between features and adaptability; they must recognize the impact of technical debt; they must invest in training, tools, and time; and they must have the commitment and discipline to deliver quality products in the face of feature pressure.

THE FINANCIAL IMPLICATIONS OF TECHNICAL DEBT

Technical debt may be costing you more than you imagined!

There has been a great deal of discussion in the agile community about technical debt. The technical debt curve in Figure 3.9 shows how technical debt increases the cost of change over time, until software becomes almost unmaintainable. Two

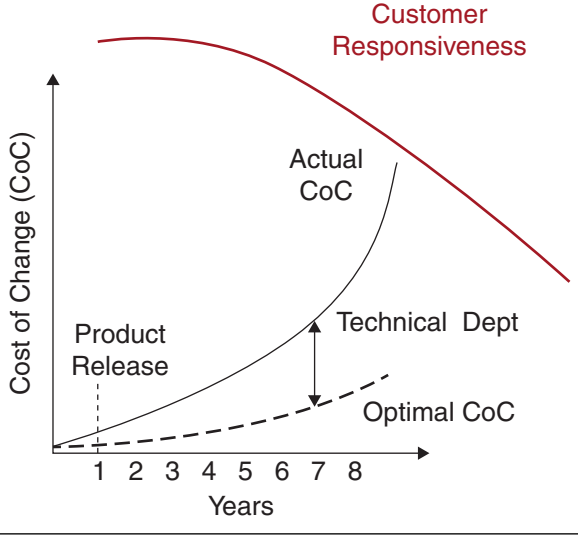

Figure 3.9 The Technical Debt Curve

time scales can be shown on this curve: a longer-term scale that shows how software degrades over time and severely impacts customer responsiveness, and a shorter-term (by iteration) scale that shows how technical debt can seriously impact development speed very quickly.

One problem with technical debt is that its impact can be slow growing and somewhat hidden. We know how the question "Fix the technical debt, or build new features?" is usually answered. As the debt gets worse, customers complain about slow delivery, increasing the pressure to take more shortcuts, which in turn increases the technical debt, which then slows the delivery process, which in turn increases customer dissatisfaction, and so on, in a rapidly downward spiraling vicious cycle. Unfortunately, by the time many organizations start paying attention, all the solutions are bad ones: (1) do nothing and the problem gets worse; (2) replace/rewrite the software (expensive, high risk, doesn't address the root cause problem); or (3) systematically invest in incremental improvement.

A number of people have been working to identify the financial cost of technical debt by examining existing software and calculating the cost of fixing bad code. Studies have indicated this overall "hidden" cost of technical debt is in the $1 trillion range in the United States. Unfortunately, this is only the tip of the iceberg when looking at the total financial impact.

Looking at the Agile Triangle (see Figure 3.1), we can envision that project value numbers (net present value [NPV] or some such measure) eventually work their way into corporate earnings. In software companies, this would be a direct correlation because the value is contained in products themselves; in internal IT projects, the correlation is a little fuzzier, but nonetheless remains valid. Quality has two components: reliability (i.e., the ability of the software to operate as billed) and adaptability (i.e., the ability to respond quickly to future enhancements). Using these definitions, technical debt can impact current value (earnings) if it doesn't perform as it should, and it impacts future earnings by slowing down the delivery process.

Unfortunately, many companies are driven by Wall Street's emphasis on current earnings as a measure of company value. When this attitude permeates the company, managers opt for delivering current features over reducing technical debt. The financial impact of technical debt then goes beyond the cost of fixing the debt, having a big impact on the ROI of any project because it delays benefits and costs more to implement.

If those are future impacts, how do we get financial managers, product managers, and others to focus on the present when it comes to reducing technical debt? The answer is seemingly simple: market capitalization.

In the fourth quarter of 2000, a well-known technology company missed its Wall Street earnings prediction by a few cents per share and proceeded to lose more than $20 billion in market capitalization in the next few days. The biggest problem with technical debt is not its impact on value or earnings, but rather its impact on predictability. Two companies with equal average earnings growth, but different earnings volatility, can have very different market capitalization driven by the volatility.

Everyone has horror stories of software applications that are virtually unmaintainable when changes cause erratic behavior. These applications are terribly difficult to test and are always buggy in production. Technical teams try to "plan" upgrades to these applications, but inevitably they miss projections, often by a lot. When a company is depending on new features to launch new product versions, delays affect earnings volatility. Getting back to the technical debt curve, as technical debt increases, development speed decreases, customer satisfaction decreases, and predictability of results decreases.

The bottom line for technical debt: it's expensive to fix, but much more expensive to ignore. Technical debt reduces future earnings, but even more critically, it destroys predictability which in turn impacts market capitalization in the near term, not in the future.

Do Less

Many managers use the mantra, "Do more with less." At the Agile 2010 conference, Pat Reed, Senior Director from the Gap Inc., shortened this mantra to "Do less." Her theme of value optimization, and eliminating marginal value work, included creating a culture of value, determining value calculations at the portfolio level, allocating value to software features, and determining the highest-value chunks of functionality to implement next—whether those chunks were projects in a portfolio or stories in an iteration planning session. By developing in an agile fashion and deploying features frequently (continuously), value can be recognized by the business early and often.

> Everyone tries to do too much: solve too many problems, build products with too many features. We say 'no' to almost everything. If you include every decent idea that comes along, you'll just wind up with a half-assed version of your product. What you want to do is build half a product that kicks ass. (The founders of 37signals (Taylor, 2011))

In an agile project, the team always tries to work on the highest-value story. But what if the highest-value story is from the next project in the development queue? Toward

the end of a project, or even earlier, the highest-value story or feature may be found in the next project, which means it may be time to stop the current project and move on.

Three studies conducted by The Standish Group (Jim Johnson, CEO, The Standish Group International, XP2002 conference) and the Department of Defense (*Crosstalk Journal*, 2002), and reported by IEEE (IEEE conference, 2002), in the early part of this decade indicate that far more than 50% of functionality in software is rarely or never used. These aren't just marginally valued features; many are no-value features. Think of the cost of these features. Think of the benefits from doing less, from eliminating these features. A CIO friend of mine once delivered a customer relationship management (CRM) application with 25% of the originally requested functionality—and the customer was delighted. In fact, the customer cut off development! The other 75% of features proved to be "nice to have" but not significant contributors to business value. Delighting customers has both a content aspect and a timing dimension. Fifty percent of the features delivered in 6 months may be far more "delighting" than 100% delivered in 18 months. Doing less should operate at all levels. The practice of allocating value to features seeks to "Do the highest-value chunk of work," but also to "Do less"—that is, to eliminate marginal valued features and cut functionality on those features with lower value.

"Do less" has other implications. Reducing work-in-process, for example, increases throughput by cutting down on time-wasting multitasking. Value stream mapping show us where to cut out non-value-adding activities. In looking at value capture, agile managers need to examine cumulative value delivered versus cumulative cost incurred on a project. Then questions can be posed, such as "Do we want 100% of the planned value for 100% of the planned cost, or would we prefer stopping at 90% of the value for 70% of the cost?" Because agile development delivers the highest-value features early, this type of management tradeoff becomes reasonable, even imperative, to think about. Furthermore, the reason this question becomes so important is the issue raised earlier—other projects with higher value need to start sooner. Developing the last 10–20% of marginal functionality on one project delays capturing the higher value on the next project. Clearly, it's not just development cost, but also opportunity cost that managers have to evaluate.

Do less: cut out or cut down projects, cut out overhead that doesn't deliver customer value, cut out or cut down features during release planning, cut out or cut down stories during iteration planning, cut down work-in-process to improve throughput. At the same time, focus on delighting the customer by frequent delivery of value. In an agile organization the mantra should be "Do less," and maybe use the time and money saved to reduce technical debt, launch new innovations, and undertake improvement initiatives.

The "To Do Less" List

"Doing less" is a deceptive term, one that gets you thinking. While you might be put off by the phrase because the mantra in most companies is to "Accomplish more," doing less actually means delivering more value, more throughput, more ROI, and more creativity, and being more focused—but doing less low-value activity.

This message is, in effect, saying, "Think more; do less." It's easy to think about what to put into a product. We often get suggestions from a wide variety of sources—engineers, customers, managers, executives, the janitor, and other teams. Product backlogs can grow exponentially. It's easier to say "yes" to various stakeholders than to say "no." One problem consistently experienced by engineering organizations is too much work-in-process (WIP). Projects are undertaken because the prioritization scheme breaks down (if there is one). It's easier to tell the vice president of manufacturing that "We're working on your project" than "We can start on your project in 2 months when we have adequate capacity." As a result, projects stack up in WIP and throughput drops, sometimes drastically, because of thrashing and multitasking.

Each of us has a handy "To Do" list, but maybe what we really need is a "To Do Less" list. There would, of course, be some process to create these lists. For example, one rule might be that for every entry on the "To Do" list, we have to make one entry on the "To Do Less" list. A product manager might create a backlog of features to do, and another backlog of features not to do. Having an explicit list of "not to do" features sends a message to everyone that this list is important. Deciding not to implement a feature isn't enough—such features have a tendency to creep back on the active pile—so documenting that a feature has been placed on the "Don't Do" list is important.

Similarly, you might keep a list of all the meetings you decided not to attend (this one isn't as difficult). As you start keeping "meetings to attend" and "meetings not to attend" lists, you begin to think about and refine the criteria for each. Actually, you will begin developing these criteria for each of your dueling lists.

You could even think about celebrating your "Not Done" lists at retrospective activities. Here are all the features we did; here are all the ones we didn't do. Here are all the meetings I didn't attend this iteration. Here are all the projects we didn't start this month. Here are all the overhead items we pushed aside last quarter. Obviously, we also celebrate what we accomplished, and we highlight that those accomplishments were sharply focused on value, on real customer solutions, and on creative ideas. By lining up both lists, we begin to see how our focus on effectiveness rather than raw productivity actually improves overall performance.

Build Less, Start Sooner

Jeff Patton reminded me of two simple strategies for software development that I've talked about from time to time—build less software and start sooner. In this section, I'll revisit these simple, but powerful strategies.

First, managers and executives complain a lot about not delivering software (or any other product, really) in a timely manner. In Preston Smith and Don Reinertsen's ground-breaking book *Developing Products in Half the Time* (Smith, 1997), these authors discuss manufactured products, not software products, but many of their ideas are relevant here. Their research pointed out that, on average, in the time frame from initial identification of a need to product ship, *more than half the project's time was taken up before the development project actually got under way!* "The front end is so fuzzy that people tend to forget that it even occurs," say the authors. They go on: "we have seen situations where as much as 90 percent of the development cycle elapsed before the team started work." How many projects with extremely aggressive schedules have you been on where everyone knows the project has been under "consideration" for months and months, if not years? Once the development team is appointed, the mantra becomes "Hurry, hurry." Where was all the hurry when management was "considering" the project?

Part of the delay in starting projects is concern about uncertainty. Because of managers' and teams' unwillingness to start before all questions are answered (and of course many of the answers will still be wrong), project start times slip—again and again. Then when the project does start—well, you know the drill. Starting early is a discipline that can greatly improve schedule performance, and at very little cost. Work on ways to get important projects off the ground early. I once worked with a medical software company in Canada that had been "investigating" a new product idea for a year. I finally convinced the company to try a few proof-of-concept iterations. The feedback was startling, but all too predictable: "We learned more about our product direction in a few 2-week iterations than we learned in the last 9 months of analysis."

One time I was asked by a senior manager in a software company, "How can you help us deliver this large product on schedule?" I replied, "Do less"—build fewer features. "Do less" is really the flip side of "Focus on what is important."

These two strategies—build less software and start sooner—sound simple on the surface, but in practice they can be very difficult to implement because of organizational inertia and politics. Even so, they can be very effective and are worth the effort to pursue.

Chapter 4

Create an Adaptive, Innovative Culture

> In the Chaordic Age, success will depend less on rote and more on reason; less on the authority of the few and more on the judgment of the many; less on compulsion and more on motivation; less on external control of people and more on internal discipline. (Hock, 1999)

Taking advantage of new opportunities requires a wide range of capabilities that are influenced by culture. While technology drives change, technology also changes rapidly. Staying ahead of the technology curve, and its business impact, requires learning, adapting, and innovating. Leaders, at any level, need the ability to adapt and the ability to lead their organizations to adapt. Changes happen to us and to our organizations. How we react to those changes and how we react to the constant flow of new opportunities will be a big determinant of success. Adapting also involves not adapting to every stimulus. Some opportunities are best left to others, and discerning which ones to advance and which ones to pass on is more art than science. As the CEO of one company joked to me, "We started the year with 12 new initiatives, now in May we've abandoned 5 and picked up 4 new ones. I don't know if we are being adaptive or are just poor planners." Welcome to the uncertainty of today's business world.

A few years ago, someone asked the question, "What would happen if we sold socks in sets of three, where none of them match?" (Williams, 2011). From this weird-sounding question emerged a new product and business, Little Miss Matched (you may have missed this phenomenon unless you have young teen girls in the house). Kodak invented the digital camera, but over many years was never able to capitalize on this new product. Disruptions come in many forms—some that must be taken advantage of quickly, and others for which the reaction timeline is longer. In either case, organizations need disruptive thinkers—those who can identify gems in the turbulence, those who can turn those gems into products, and those who can shepherd those gems through the corporate political system. All of the practices and processes related to managing the flow of opportunities will be for naught if an organization doesn't have the disruptive thinkers to breathe life and vitality into those opportunities.

The single greatest barrier to effective agile/Lean transformations is focusing on practices and ignoring the changes in mindset that come from embracing agile and Lean values and principles. Behaviors are more ingrained—it's difficult to learn to be more empathetic. It's not easy, but is somewhat easier, to learn to be more adaptive.

> A traditional manager focuses on following the plan with minimal changes, whereas an agile leader focuses on adapting successfully to inevitable changes. (Highsmith, 2010)

Extensive lists of mindsets have been identified with adaptive leaders, but four core concepts are Adapting, Exploring, Engaging, and Riding Paradox. This chapter focuses on these four leadership traits.

ADAPTING

Change is inevitable, but what we can manage is how we respond to change. In an environment of volatility, ambiguity, and uncertainty, how can leaders expect to conform to a plan, in particular one that predicts results a year or more in the future? While most managers would agree that change is inevitable, those same managers often fail to put appropriate adaptation mechanisms in place.

> Adaptive leaders emphasize "articulating goals, facilitating interactions, improving team dynamics, supporting collaboration, and encouraging experimentation and innovation. (Highsmith, 2010)

Most organizations (of any size) encourage conformity and optimization. To be agile and adaptive, we need to encourage risk taking and quirkiness. Having an adaptive mindset means that someone is open to change and understands the change process—opening individuals to see reality as it is, not as they think it should be; realizing that adaptation is a natural process that can be goal directed, but not controlled; grasping that adaptation is driven by emergent (innovative) results that are generated by collaborative processes operating at the edge of chaos (minimal structure); organizing for rapid decision making; and acting for change.

Having an adaptive mindset also means understanding the change process, including how people and organizations are apt to change, and how they are apt to resist change:

> Experimentation matters because it is through learning equally what works and what doesn't that people develop great new products, services, and entire businesses. But in spite of the lip service that is paid to "testing" and "learning from failure," today's organizations, processes, and management of innovation often impede experimentation. (Thomke, 2003)

Today's organizations, processes, and management often impede successful adaptation to business turbulence. Changing and adapting are not the same, and the difference between them is important. There is no goal inherent in change—as the quip says, "Stuff happens." Adaptation, in contrast, is directed toward a goal (suitability). Change is mindless; adaptation is mindful. Adaptation can be considered a mindful response to change. Success asks us to alter our mindset to "embrace change" (Beck, 2000) by being "focused, fast, and flexible" (Horney, 2007) and by using appropriate models. It's one thing to say, "Be adaptive"; it's an entirely different matter to offer leaders concrete practices or models to assist them.

In the natural world, mutation and natural selection drive adaptation. Mutation provides choices, most of which are rejected; natural selection picks the winners and losers. In the business world, opportunity and innovation provide the choices, and competitors and customers pick the winners and losers. So leaders need to have ways of doing both—generating lots of innovations (mutations) and choosing potential winners.

> I suspect that the fate of all complex adapting systems in the biosphere—from single cells to economies—is to evolve to a natural state between order and chaos, a grand compromise between structure and surprise. (Kauffman, 1995)

Anticipation (planning) and adaptation aren't the antithesis of each other, but rather are complementary—you have to do both. Failure to anticipate and plan leads to unnecessary rework and possible failure. Conversely, trying to anticipate the unknowable leads to unrealistic plans and expectations. Plans evolve from what we know. Adaptations are responses to learning what we don't know. Flexibility is the response to changes we expect in the future. For example, we know that tax rates will probably change in the future, so we build flexibility into our processes and software. Adaptability is how we respond to the unknown—it's maintaining structural quality (architecture, infrastructure, process) and enterprise agility. Unfortunately, it's not always clear what we know and what we don't.

Adapting requires new mental models:

> As quantum physics changed our notions of predictability, and Darwin changed our perspective on evolution, complex adaptive systems (CAS) theory has reshaped scientific and management thinking. In an era of rapid change, we need better ways of making sense of the world around us. Just as biologists now study ecosystems as well as species, executives and managers need to understand the global economic and political ecosystems in which their companies compete. (Highsmith, 2010)

The four models discussed here can help organizations become focused, fast, and flexible, but they are merely starting points. Leaders need to adapt these models to their unique situations, or find useful substitutes. These four models that can assist in learning how to adapt are the Purpose Alignment, Satir Change, Short Horizon, and OODA Loop models. They help us answer four key questions:

- What is important?
- What is our model for change?
- What time frame are we working in?
- How do we make fast, effective decisions?

The next sections introduce tools that address each of these questions.

Purpose Alignment Model

The first question to ask when responding to turbulent change is "What's important?" This question is easy to ask, hard to answer. One effective tool for doing so is the Purpose Alignment Model (Pixton, Nickolaisen, Little, & McDonald, 2009), shown in Figure 4.1. This model can be used at any level—strategy, project, and feature. Its two dimensions are market differentiation (Does this really make a difference?) and mission criticality (Is it something we have to do to succeed?). For example, in most businesses, billing is mission critical (must be done) but not differentiating (having the best billing system won't scare the competition). Usually you only need to match competitors' billing systems (parity). In looking at potential adaptation initiatives, it's important to first ask why and how it matters.

One of the hardest things leaders do is choose. So many options are available today: this product, that product; onshore, offshore; bricks and mortar, Internet; extensive

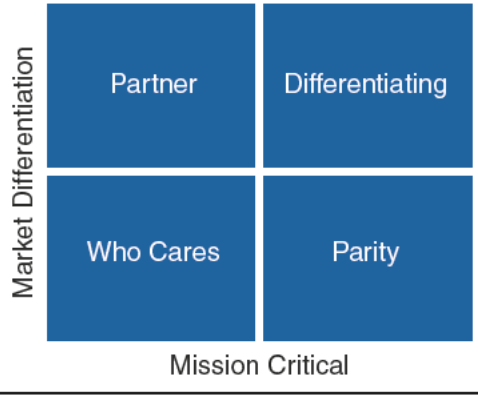

Figure 4.1 Purpose Alignment Model

marketing, word-of-mouth marketing; social networking, traditional networking; data center, cloud—the list of possibilities is endless. One of the things that distinguishes effective leaders from ineffective leaders is choosing well. Models such as the Purpose Alignment Model can help, but they don't make decisions, leaders do.

Satir Change Model

The Satir Change Model shown in Figure 4.2 is one of a number of useful ways to think about change (Weinberg & Smith, 2000). I like this model because it emphasizes some key points:

- Things get worse before they get better.
- People may give up on a change if it gets too uncomfortable.
- The ride from current performance to better performance is bumpy.
- Successful transitions require investments in both time and money.
- Trust and understanding are needed to overcome fear and resistance.

This model, and many others, seems to ignore the question, "Is this a good adaptation?" The entire process is geared toward overcoming resistance, and resistance

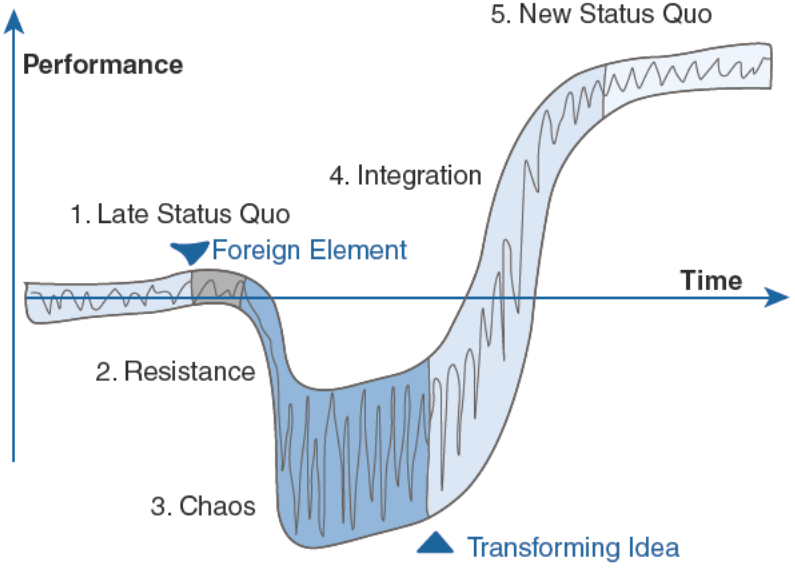

Figure 4.2 Satir Change Model [Illustration adapted from Steven M. Smith's "The Satir Change Model," published in *Amplifying Your Effectiveness: Collected Essays*, G. M. Weinberg, J. Bach, and N. Karten, eds. (New York: Dorset House Publishing, 2000), p. 96. Copyright © 2000 by Gerald M. Weinberg. Used by permission of Dorset House Publishing (www.dorsethouse.com).]

always has a negative connotation. But think about change for a minute. Environmental changes create both opportunity and danger. In any business, development organization, or project team, there are many, many changes: market, economic, competitor, team member, business objectives, and so on. For any one of those changes, there may be multiple possible adaptations. With hundreds of changes, large and small, and hundreds of possible adaptations to each, again both large and small, how do we weed out the adaptations that are wrong choices?

Maybe we should look at the Satir model, and others, not just from the negative perspective of overcoming resistance, but also from the positive perspective of helping to weed out the inappropriate adaptations while helping us implement the appropriate ones. We tend to think of change management, perhaps better called adaptation management, as managing the exceptions—the deviations from the norm. But maybe we should view adaptation as the normal and a steady state as the exception—it sure seems that way in today's business environment.

Short Horizon Model

Managers and executives tend to think in certain horizons—strategic, tactical, and operational. However, in a turbulent world the traditional time frames for these horizons (a year for an operational time frame, for example) are too long. A better Short Horizon Model for responding quickly to opportunities and threats is the roadmap, release, and iteration model used by software delivery teams, shown previously in Figure 3.6. Business initiatives can be planned and executed with this model. A roadmap (or visionmap) puts large chunks of work onto a 6- to 18-month timeline. Within the roadmap, release plans, consisting of deployable chunks of work, are outlined in a 3-month timeline. And at the lowest level, 2-week iterations, consisting of small, useful chunks of work, are planned within each release. If executives and managers want to be adaptive, then they must shorten their working cycles just like agile software deliverers do.

OODA Loop Model

The fourth useful model in building adaptive mindsets and organizations is the OODA Loop developed by U.S. Air Force fighter pilot John Boyd. Boyd was an ace fighter pilot and had great influence in fighting strategy. His OODA Loop—Observe, Orient, Decide, Act—shows the thinking process behind making lightning-fast actions and responses to a competitor's action.

The way the basic OODA Loop Model (using simple arrows around in a circle) is normally depicted is somewhat deceptive, because the fast-and-normal path is actually OOA (Observe, Orient, and Act)—the way it's depicted in Figure 4.3. For really fast action, Boyd depended on training and experience guiding him directly to action,

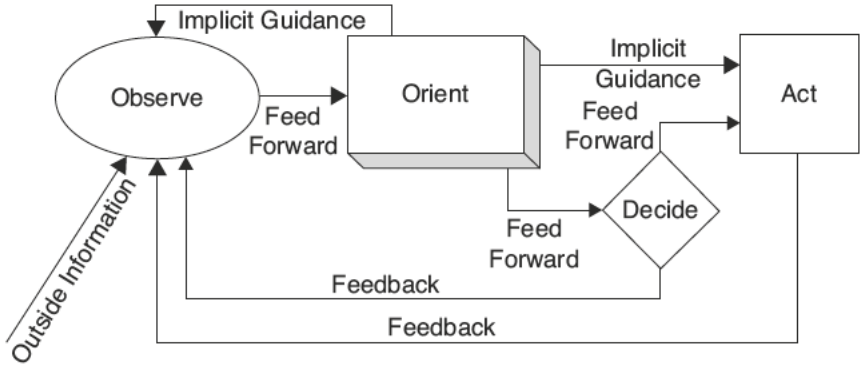

Figure 4.3 OODA Loop (Adapted from Boyd, 1995.)

without a lengthy decision step. The decision step was usually performed after the fact, acting as a learning practice. Boyd also differentiated between observing and orienting—the former was seeing reality without filters, while the latter applied the filters of culture, experience, new information, and analysis. In a turbulent environment, the importance of seeing reality without filters enhances the ability to identify opportunities and threats.

Pivots and Adaptations

An interesting concept emerging from the Lean Start-up movement is the pivot. A pivot occurs when a start-up company realizes its business model or new product isn't working in some significant way, so it changes or pivots to a revised model. While the "pivot" term has seemingly been overused and overhyped, this concept can be applied to projects. Some project changes are normal adaptations—that is, smaller changes to accommodate the myriad of things that arise during a project. Occasionally (not more than once or twice a project, one hopes), large issues arise that entail a significant change in the project's value proposition, a major problem with the technology or architecture, or an updated release plan that indicates major schedule problems. At this point, the project should "pivot." Pivoting entails a recognition that at project or product plan won't work as envisioned. It should be viewed as adapting to changes or learning new information, not as an opportunity to start the blame game.

Don't Plan, Speculate

One thing people are learning is that you can't plan uncertainty away. Plans are good for things we know, or things that we may have some control over. However,

uncertainty—along with its close cousins ambiguity and vagueness—defies planning. When I originally introduced my Adaptive Life Cycle in *Adaptive Software Development* (Highsmith, 2000), the three high-level phases were Speculate, Collaborate, and Learn. I expanded on these in *Agile Project Management* (Highsmith, 2010) to be Envision, Speculate, Explore, Adapt, and Close. In either case, Speculate takes center stage.

Planning says, "I know something and this is how it's going to be." Speculate says, "Here is my hypothesis about the future, let's explore."

> In a complex environment, following a plan produces the product you intended, just not the product you need. (Highsmith, 2000)

In a traditional approach, deviations from plans are mistakes that must be corrected. In an adaptive approach, deviations guide us toward the correct solutions.

When we speculate, we define a mission to the best of our ability, but our choice of words indicates that we are more than likely wrong. The word "speculate" first calls to mind the image of reckless risk taking, but actually the dictionary definition is "to conjecture something based on incomplete facts or information." I must admit that some of my early banking clients had a hard time using a development phase named Speculate, but somehow we need to change the tenor of discussions around what is traditionally known as planning. "Planning" is too deterministic a word, in that it conjures up a mindset of fixed time frames and detailed activities. Speculate says, "Within our product visions, let's develop a hypothesis about the future and then test it in short iterations with our customers."

In *Experimentation Matters: Unlocking the Potential of New Technologies*, author Stefan Thomke (2003) focuses on the need for experimentation as a driver for innovation: "But in spite of the lip service that is paid to 'testing' and 'learning from failure,' today's organizations, processes, and management of innovation often impede experimentation." As the sentence warns, while many managers talk about learning, their organizational systems impede that learning. We need to stop planning and begin speculating, hypothesizing, and experimenting.

I was slow to embrace the term "Story" for iteration planning. What finally convinced me was this need to change people's perception of planning. The traditional terms we used were "requirement" and "requirements document"—words that conjure up fixed, unvarying, cast-in-concrete outcomes. "Story" conjures up a different scenario, one that emphasizes talking over writing and evolution over a fixed specification.

Using the term "speculating" rather than "planning" has a similar effect. When we speculate, we are not prescribing the future, but rather hypothesizing about it. And, to those who practice the scientific method, we hypothesize and then run experiments to test that hypothesis—exactly what happens in agile iterations. Speculating also conjures up a vision of a group musing about the future instead of one rushing to document in detail. Words make a difference; they convey a nuance about how to conduct ourselves. So I'll make another pitch for elevating Speculating and demoting Planning.

Embracing Change

To a manager's question "How much will project Zebra cost?", the comment "I don't know" is usually an unacceptable (but often the best) response. A more specific answer, such as "Well, somewhere between $2 million and $6 million," would be rejected in many, if not most, organizations. The project manager who says, "Project Zebra will cost $3.2468 million," will be hailed as "with it." Now, everyone knows down deep that this figure is fantasy, but they are comfortable with it because they know (1) there is an off chance it will be right or (2) it will be wrong but we can deal with that when the time comes. People seem to be more willing to accept a highly questionable figure over a range of numbers that delineate the degree of uncertainty.

One of the principles of the Declaration of Interdependence (DOI; www.agileleadershipnetwork.org) addresses the topic of uncertainty directly: "We expect uncertainty and manage for it through iterations, anticipation, and adaptation." An approach to uncertainty must be multifaceted—no single strategy or practice will be enough.

Having an agile mindset means accepting uncertainty about the future as a way of dealing with the future. Any development effort should be a balance between anticipation (planning based on what we know) and adaptation (responding to what we learn over time). Managers, and teams, need to find the balance point—how much time we spend investigating requirements, technology, and other issues in the beginning of a project versus actually developing product features (working software) and adjusting/adapting to new information as the project unfolds.

Traditional teams attempt to drive out uncertainty by planning and analysis. Agile teams tend to drive out uncertainty by developing working software in small increments and then adjusting. Traditional teams often think they have mitigated much of the uncertainty, when in fact a high degree of uncertainty still remains in the project. Late in the project the uncertainties begin wreaking havoc—and major cost overruns and schedule slips are announced.

Agile teams, in contrast, accept uncertainty early in the project but become more certain later in the project. Agile project managers are often hard pressed to defend their uncertainty because upper managers possess the "You can be right, and you can be wrong, but don't be uncertain," mindset. Dealing positively with uncertainty, being willing to accept that certain things are unknown and unknowable (for now), is a big part of learning to be an agile leader.

Change Isn't Change

It seems as though everyone now prefaces presentations, blogs, and books with dire predictions about change in the world. What we fail to recognize at times is that change isn't change, but rather there are different types of change and we need to have different tools—different levels of innovation—to address each. Three types that come to mind (there are surely other useful classifications) are incremental, differentiating, and disruptive.

Much of the industry discourse on change, and the innovation to respond to it, reminds me of a story told by the head of a teachers organization in Utah years ago. The person sitting beside her on a flight asked, "What's the one thing we need to do to fix education?" She replied, "Get rid of all the people who think there is one thing that will fix education." The parallel here is that how we approach change should be multidimensional and contextual. Any answers we come up with have to fit a particular context, within a certain period of time, until that context changes. There isn't one answer.

Incremental change includes small new items added to an existing product, practice, or process. They are the small "ah ha" moments that lead to improvements. Teams innovate in this way all the time as they listen to customers and think up new features.

Differentiating changes involve a significant product, practice, or process enhancement that creates a clear differentiation from the competition and that keeps us ahead of the competition for some short-to-medium time frame. Innovating to respond to this type of change is difficult because it creates differentiation within a market, but doesn't significantly change a market.

Finally, the toughest change to respond to is disruptive change, which requires innovating to create an entirely new market or change the market economics substantially. Disruptive changes are those that cause companies to fail if they don't respond appropriately—and that response has proved to be particularly tricky to carry off for many firms. Most disruptive changes have been classified that way only considerably after the fact. Although these disruptive changes were actually manifested over a period of years (think the demise of minicomputers), companies just couldn't fathom the disruption even as it was happening to them.

Each type of change requires innovative responses, and each also requires a different level of innovation. Obviously, it gets much harder as we move from incremental changes to differentiating changes to disruptive changes. As you design an opportunity management program for your organization, try to think about the different types of changes and ask which innovation practices fit best with each one.

Continuous Change

All of us fall into clichéd phrases about the constancy and pervasiveness of change—yet our conceptual framework and management practices still view change as an exception. Most of our concepts, practices, and tools are geared toward environments in which equilibrium is the normal condition and changes are the exception. Extreme, high-speed, high-change environments are just the opposite—there, change is the norm.

> In extreme environments, equilibrium is the exception condition. (Highsmith, 2000)

In such circumstances, change management is not an add-on procedure to a linear process temporarily in disequilibrium, but rather the heart of leadership. Adaptation is the process of continuous change, and stands in sharp contrast to the periodic discrete change process found in many organizations. The difference between beginner and expert skiers provides an analogy. The beginner traverses the slope until he or she encounters the trees at the edge. For skiers, trees provide a significant incentive to change (turn). However, the expert skier is always changing, always turning, always on edge, always adapting to the challenge down the hill. The beginning skier is more like a traditional organization, utilizing existing practices until the threat (or actuality) of encountering trees is so great the skier has to change. Adaptive organizations, like advanced skiers, treat continuous change as the norm—and their practices reflect that actuality.

The analogy has a further dimension. Watching beginning skiers is often painful; their arms are flailing, their skis cross intermittently, their bodies struggle for balance. Expert skiers flow down the mountain, their arms barely moving for the next pole plant, their skis appearing fastened together, their upper bodies hardly moving.

Some organizations lurch from change to change. They fight change even as they try to manage it, flailing away and expending tremendous energy, while other organizations seem to gracefully absorb the inevitable bumps along the way.

Adaptive change is more graceful because it flows from all levels of an organization. Historically, however, change has been imposed in a top-down manner. From Newtonian determinism came Frederick Winslow Taylor's *Principles of Scientific Management*, first published in 1916, which in turn spawned an almost slavish focus

on process and workflow. Early science and early warfare laid the foundations for a command-and-control management philosophy: the manager knows the objective and commands the troops to conquer the objective; once the command is given, the manager monitors progress and controls the outcome. This approach worked well as long as the objective to be conquered did not move around much, and as long as the organization existed in a more predictable world.

The Emotions of Change

The agile movement has encouraged people to change—from changing technical aspects of programming, to changing social interactions via increased collaboration, to changing management styles to be adaptive and facilitative rather than focusing on command and control. Sometimes in thinking about organizational change, we forget that it is ultimately a "people" change and that organizational change is the accumulation of individuals who are changing. At an individual level, change is first and foremost emotional.

On the positive side, change holds out the promise of exhilaration, hopefulness, excitement, enthusiasm, optimism, wonder, rejuvenation, and surprise. On the flip side, change can bring out feelings of suspicion, fear, foreboding, frustration, anger, dislike, confusion, hesitation, numbness, disquiet, uneasiness, fatigue, insecurity, anxiousness, irritability, and being stressed out.

The places we are changing from are rarely completely bad, and the places we are changing to are rarely all good. This ambiguity often makes the changes difficult because it intensifies the emotions surrounding questions like "Are we really making the right choice?" We need to be more cognizant of the fact that successful change involves both analytical planning and managing the reality of emotions.

"Agile," "flexible," "adaptive," and "dynamic" are all words we use to encourage people and organizations to be more responsive to the turbulence in our business and economic environment. We then deride those whom we view as resistant to change. But maybe what holds us back is as much "fear of choice" as it is "fear of change." Author Seth Godin (2010) says that entrepreneurs and leaders who create tons of value for their companies often perform the same tasks as the rest of us, except for a critical 5 minutes per day. In that 5 minutes, they are somehow able to cut through the thousands of possible choices and select the one that creates value. For others, this cornucopia of choice is paralyzing; it's the fear of choice that holds them back.

All the 4 × 4 matrix models in the world won't make the key decisions for you. Models assist, but ultimately individuals and core teams cut through the profusion of choices—go left or go right, go north or go south, build product A or build product B. In Lewis and Clark's famous trek across the mountains and plains of the Northwest to find the Pacific, there was a particular key juncture. Nearly everyone besides

the leaders thought the right fork was the Missouri River, but the leaders chose the left—the correct one. Maybe we don't need more books on change management; maybe what we need is a few good ones on choice management.

EXPLORING

Adapting is about understanding the fundamental process of seeing reality, embracing change, and responding to that change. Exploring is similar, but more about the "how" of that response—an explorer knows how to experiment, how to learn and evolve a solution over time. Leaders who adapt are comfortable with responding to unfolding conditions and changes. Leaders who explore use an Envision–Evolve process rather than a Plan–Do process. One aspect of agile software delivery that sounds easy but in reality has proved quite difficult for many individuals to accept has been the idea of gradual evolution rather than a big upfront effort (be it design, requirements, architecture, or business model). The idea of creating a skeleton architecture or skeleton plan and having it evolve over time, rather than doing extensive data gathering upfront and then issuing a final plan or an architecture, just seems foreign to many. In many ways it feels like a loss of control, which it is to some extent, but what people don't realize is that they never had control in the first place. It's interesting that many managers and leaders are comfortable with a prescriptive plan, even when they know that historically these plans haven't worked out and the results will probably be different. Even so, they are uncomfortable with a fuzzy early plan that evolves toward a goal.

> To create, a person must have knowledge but forget the knowledge, must see unexpected connections in things but not have a mental disorder, must work hard but spend time doing nothing, must create many ideas yet most of them are useless, must look at the same thing as everyone else yet see something different, must desire success but learn how to fail, must be persistent but not stubborn, and must listen to experts but know how to disregard them. (Michalko, 2010)

Changing the Plan–Do management culture won't be easy. An Envision–Explore culture understands that innovative answers to complex problems emerge over time. This idea of letting solutions emerge rather than having them be predetermined upfront in the plan takes a leap of faith for many managers—they want to know the precise steps from here to there. They are uncomfortable with a process that says, "Let's plan a little, get started, and we'll see what happens." They want answers where there are none. They are comfortable with a detailed plan, which they know won't work out, but offers the illusion of a known end point.

Even more difficult may be that the process doesn't really involve choosing between one way or the other. It's not plan or don't plan; it's create a vision, plan some, execute

some, and plan again. People, not just managers, want certainty in a world of uncertainty. Those who can learn to deal with, and tolerate, an Envision–Explore mentality will have a higher success rate in responding to today's challenges.

ENGAGING AND INSPIRING

Transforming an organization so that it embraces enterprise agility involves everyone, from delivery staff to project managers, to functional managers, and to executives. One of the key leadership tasks is to inspire others to achieve the goals of the transformation and to effectively engage them in the process. According to a Towers Perrin Global Workforce study published in 2007–2008 (Perrin, 2007), "Only one in five of the global workforce is fully engaged." One in five isn't enough to power a critical transformation.

The agile community has focused on engagement and motivation by advocating self-organizing teams, collaborative interactions, technical excellence, participatory decision making, and adaptive (not command–control) leadership. *Drive*, by Dan Pink (2009), takes a look at the research that supports and enhances the agile approach to engagement. Pink says that management has been ignoring research into motivation for many years. The motivation myth is that more stuff (extrinsic factors such as money) yields more productivity. The research shows that except for routine jobs that require very little cognitive ability, motivation is actually driven by intrinsic factors: autonomy, mastery, and purpose.

> Human beings have an innate inner drive to be autonomous, self-determined, and connected to one another. And when that drive is liberated, people achieve more and live richer lives.

Research gathered by Pink backs up his claims: "for example, researchers at Cornell University studied 320 small businesses, half of which granted workers autonomy, the other half relying on top-down direction. The businesses that offered autonomy grew at four times the rate of the control-oriented firms and had one-third the turnover." In the agile community, this need for autonomy has been met by the push to implement self-organizing teams, teams whose members have a degree of autonomy over how they work and how decisions are made. Intrinsic needs come from within; they convey belonging. Extrinsic things come from without; they are the result of external forces. When a manager attempts to motivate a person or a team, he or she is trying to influence behavior by offering incentives. When a manager attempts to inspire a person or a team, he or she is trying to influence behavior by offering purpose.

Money is an incentive. Saving the environment is a purpose. Inspiration is bigger and longer-lasting than extrinsic motivators; it speaks to the heart as well as the head.

Leaders can inspire others to greatness, but they can't motivate them to it. The historical context of motivation tends to convey a message emphasizing its short-term nature, rewards, narrow focus, and control. Inspiration tends to convey a message that is longer term, emphasizes internal feelings of satisfaction, broader purpose, and wider focus, and is visionary rather than controlling. As agile leaders, at all levels of an organization, we should strive to inspire rather than to motivate, to engage people in the transformation process. If we want people to be innovative and creative in coming up with new products and services, we need to inspire them to greatness rather than motivate them to mediocrity.

Facilitating

Traditional command–control management is about managers telling subordinates what to do, when to do it, how to do it, and where to do it. There is little autonomy in this kind of hierarchical culture. Adaptive leaders, by comparison, are more facilitating than demanding. Their job is to create a self-organizing, self-disciplined team—whether the team develops software or manages the business. Effective leaders are increasingly collaborators. One survey posed the statement, "You have programs designed to develop leaders who can creatively bring together resources across different parts of your organizations." In the top 20 performing companies, 100% of respondents agreed with this statement. In all others, the agreement was 66% (Hay Group, 2010). Collaborative leaders run the top companies. Adaptive leaders lead teams; nonadaptive ones manage tasks. How many managers spend hours detailing tasks into Microsoft Project and then spend more hours ticking off task completions?

Many managers like this task-oriented approach because it is concrete and definable, and because completion seems finite. Facilitating teams, in contrast, seems fuzzy, messy, undefinable, and never complete. Naturally, some people gravitate to the easier path—managing tasks. Adaptive leadership focuses on team management, from building self-organizing teams to developing a servant leadership style. It is more difficult, yet ultimately more rewarding than managing tasks. In an agile enterprise, the people take care of the tasks and the leader engages the people. The facilitative leader works on things like building self-organizing teams, creating a trusting and respectful environment, ensuring collaboration, encouraging participatory decision making, and developing appropriate empowerment guidelines (for an excellent discussion of empowerment, see Chapters 6 and 7 in Appelo, 2011).

> Commanders know the objective; leaders grasp the direction. Commanders dictate; leaders influence. Controllers demand; collaborators facilitate. Controllers micro-manage; collaborators encourage. Managers who embrace the leadership-collaboration model understand their primary role is to set direction, to provide guidance, and to facilitate connecting people and teams. (Highsmith, 2000)

Facilitating a collaborative, self-organizing organization may be the most important job of an adaptive leader. Of course, being a facilitative leader doesn't mean abdicating all authority and decision making. Another primary task for an adaptive leader is bringing clarity to ambiguous situations. Clarity sounds simple, but it's not. We embrace agility because it helps us adapt to the turbulence that creates opportunity and peril. Most of the time significant changes create mounds of uncertainty and the decisions required to respond to those changes are never clear-cut. There is never one obvious option, but rather a multitude of options that seem reasonable. There is never enough information, and what information exists is often contradictory. Change creates ambiguity, uncertainty, doubt, and indecision that can lead to floundering. Adaptive leaders have the ability to cleave through this ambiguity, to focus on a decision when everyone else is floundering, to clarify direction when everyone else sees confusion. In today's highly amped environment, waiting for certainty ensures failure. In an article in *Harvard Business Review* several years ago, a CEO of a fast-moving, high-tech company said something to the effect of "my job is to reduce ambiguity." He realized that at some point the debate among his management team needed to end, that at some point he needed to cut through the uncertainty and make critical decisions. He needed to be clear, even when everyone knew the situation was uncertain.

Adaptive leaders are those who have vision and foresight; who can articulate clear direction; who can persist in the face of ambiguity, uncertainty, and doubt; who can adapt before their focus becomes obsession. Growing leaders who embody these traits is a critical task in building agile/Lean organizations.

> The structure of an organization's collaborative network has significant impact on its ability to produce emergent [innovative] results and ultimately on its very ability to adapt. (Highsmith, 2000)

> The Top 20 Best Companies for Leadership are at the forefront of a significant shift away from hierarchical organizational operating models. (Hay Group, 2010)

Oscillation versus Iteration

Short iterations can cause agile teams to lose focus and begin oscillating rather than iterating. This can happen from several perspectives—business, technical, user—and it's something that agile teams need to be aware of and guard against. When customers change their minds on a user interface issue over and over again, oscillation may be the issue. When developers skip from one design to another over and over, oscillation may be the issue. When product owners churn stories into and out of iterations, oscillation may be the issue.

Iterations converge on a completed story, feature, or objective. Oscillations send us back and forth between states with little forward progress. It's difficult at times to tell the difference, but when an oscillating situation seems to be happening, someone needs to stand back and say, "Stop." This is the time to stop and reevaluate the situation. Two typical problems are (1) the participants don't understand the problem or (2) the participants have different opinions about the solution. In either case, continuing on the same path won't resolve the issue.

Often the problems arise from not having a well-defined context for the work—for example, not having a good theme for an iteration or a visionmap for a release. When discussions about a particular story (how much functionality it should deliver, for example) seem endless, then perhaps the team doesn't have an anchor point in terms of what they were trying to accomplish during the iteration. When two developers argue endlessly about a design approach, maybe the team failed to set basic design guidelines.

Teams without release plans often oscillate in iteration planning because they lack the overall theme and context that such a plan provides. I once worked with a cell-phone software team during a particularly volatile time in the industry. Standards were changing and manufacturers were changing their requirements frequently. The team felt that they were going in circles—and their solution was to freeze requirements. Unfortunately, that approach would have been disastrous in the market. This was a case in which changes, while seemingly oscillations, were just normal for the industry at that time. The company did need to adapt to these changes and iterate toward a solution. By going back to the business objectives for the project, in this case the multitude of changes were justifiable.

Separating oscillation from iteration isn't always easy, but it's a skill that good Scrum Masters, project managers, iterations managers, and teams need to cultivate.

Micromanaging Angst

I've always been concerned that some agile practices are applied even when they are not appropriate for a particular situation. I've called this Agile 101, referring to learning the basics, what Alistair Cockburn calls the Shu level of learning (Shu-Ha-Ri are the three levels). All too often, some agile practice or misunderstood principle will be inappropriately used.

One of the tenets of managing self-organizing teams has been the agile mantra, "Don't micromanage." The question in my mind is whether this always applies. Agile leaders are admonished (I've said this myself) to create a clear vision, establish appropriate boundaries, and facilitate team collaboration—and then get out of the way!

There are certainly circumstances when this style of management is appropriate, but is it always?

What got me thinking about this paradox of micromanagement versus macromanagement was reading Walter Isaacson's (2011) biography of Steve Jobs. Jobs was in many ways a micromanager of extreme proportions. When it came to product development, he inserted himself into excruciatingly detailed design decisions. There was seemingly no in-between with Jobs—it was either wonderful or crap. Product review meetings with Bill Gates were also legendary—and he was noted for uncovering the slightest flaw in thinking about new products.

So would Jobs have been categorized as a good agile leader? If not, are we offering people the right agile leadership model? There is no doubt that Jobs built one of the most successful technology companies ever. Gates built another. This appears to be one of the paradoxes of good management, and whenever there is a paradox, or a seeming one, looking at it from a different angle often helps.

If we take a close look, there appear to be some interesting differences between forms of micromanaging. Jobs focused on product, not process (at least based on the information in Isaacson's book). Jobs learned from his father to focus on every aspect of the product, even the insides that were hidden from view. He obsessed about every detail, often driving his teams to try version after version after version (iterative design in the extreme). In many ways, Jobs played the roles of product owner and customer, not a manager. While he obviously had managerial authority over the staff, and used this authority, his passion was about creating great products. There is little doubt that Jobs's passion about products made Apple into a stellar business success.

When we think of micromanaging, meaning the "bad" kind, we are usually thinking about it in the context of managing the "how," the detailed list of tasks and endless focus on process. While adaptive, agile leadership should focus on vision, engaging, and boundaries. At the same time, it should be about "product" management or leadership—meaning that it should be about creating outstanding, innovative products. The first part is macromanagement, the second micromanagement.

When we talk about micromanaging, we need to separate the "bad" kind from the "good." That also raises the next question, "Is there some other good to be extracted from the bad?"

Can-Do Thinking Makes Risk Management Impossible

> Can-do thinking makes risk management impossible. Since acknowledging real risk is defeatism, the risk management function in a can-do organization is restricted to

> dealing with those smallish risks that can be mitigated by quick action. That means you confront all the risks except the ones that really matter. (Tom DeMarco, *Why Does Software Cost So Much?*, 1987)

Of the many, many quotes from Tom DeMarco that I've used over the years, this one is near the top of my favorites list. I'm reminded of a major airline that launched into a comprehensive—hotel, car rental, and so on—reservation system long ago. After a $100 million expenditure, the project was cancelled and a half-dozen project managers were fired for "lying" about progress. I have always wondered whether the company "culture" forced the project managers into an untenable situation in which "shading" the truth became the only escape.

"We're going to be over budget by about 10% and will need an additional 4 months to get this project finished," states the project manager in a status meeting. "Unacceptable, fix it, make the numbers work out!" the executive in charge shouts. "If you can't, I'll get someone who can," he rages as he stalks out of the room. Now, as a project manager, even the most dedicated and ethical one, how many times will you be willing to endure this kind of tirade? Who is at fault here—the project manager who begins to shade the truth of a project's progress, or the executive who refuses to listen to reality? I can imagine a similar conversation going on in the halls of Enron.

In working on a project plan at a major software developer, a young, gung-ho project manager responded to a suggestion to develop a risk management plan. "We don't need a risk management plan," he emphatically stated, "because this project can't be allowed to fail."

Project teams and project managers are loath to admit to uncertainty and not knowing, because the prevailing culture in many organizations doesn't allow for it. Voicing concerns is tantamount to negativism; concerns are seen as a failure of willpower, as if will alone could overcome all obstacles. Again, it seems that Enron's executives suffered from the hubris of thinking that they could make anything happen by sheer force of determination and will.

Risk management is a critical part of good project management, as experts like Tom DeMarco vividly explain. But effective risk management begins with a culture in which reality isn't overwhelmed by hubris. No project manager can be a good risk manager in a political environment where executives are so gung ho that it blinds them from reality. Effective risk management requires that executives and project managers make hard tradeoff decisions as the speculative hypotheses we call "plans" are smashed against the harsh reality of the rough-and-tumble world. If the largest companies in the country can't "control" events, it's unlikely that your project team will be able to, either. When actual performance doesn't conform to the plan, and

when unforeseen (or foreseen, for that matter) risks attack your project, the ability and will to acknowledge reality and make good tradeoff decisions will ultimately determine success.

Risk management and politics go hand and hand. Trying to implement sophisticated risk management practices in organizations that refuse to accept reality is an exercise in futility.

Making Self-Organization Work

> "Discipline without freedom is tyranny; freedom without discipline is chaos."
>
> —Cullen Hightower

Morning Star is the largest tomato processor in the United States, with 400 employees and more than $700 million in annual revenues. Morning Star's CEO, Chris Rufer, built a successful company on the principle of self-management. At Morning Star, everyone is responsible for coordinating with colleagues, customer, and suppliers, absent directives from others. (This story comes from management guru Gary Hamel's recent book, *What Matters Now: How to Win in a World of Relentless Change, Ferocious Competition, and Unstoppable Innovation* [2012].)

What struck me about this story, which is essentially about moving from a hierarchical to a networked management structure, from command–control to self-management, were the success, the benefits, and the difficulty. Self-management, or self-organization, doesn't come easy. It's so much more than saying, "Okay, now everyone is empowered." It takes three distinct things to make self-organization work: values, accountability, and systems. All three of these things are embedded in the Morning Star culture, and they make the company's success difficult to duplicate.

First, the values of self-management run deeply in Morning Star. Every employee receives training in this from the beginning, and new hires who have experience in more traditional companies often have significant problems making the change. For example, there is no centralized purchasing group; if an employee—any employee—needs something, that person buys it!

Networks are built on peer-to-peer agreements, not top-down plans. Each Morning Star employee and business unit develop individual mission statements that establish the context for their work. Then, each individual and business unit develop a set of "Colleague Letters of Understanding" (CLOUs) that outline how each will work with others. (I proposed something similar in *Agile Project Management* called an Interteam Commitment Story that helps in managing large projects.) These CLOUs

include activities and relevant performance metrics. As Chris Rufer says, "The question is not whether you have structure, but how you develop it—top down or bottom up." The important point here is that these are peer-to-peer agreements, not employee-to-boss commitments.

Agreements, of any kind, aren't effective without accountability. At Morning Star, accountability is built using a number of systems, but eventually it boils down to individual commitment and culture. Anyone who works at Morning Star has the authority to spend money and hire new employees, but they must justify those acts to their peers. Consequently, the bigger the decision, the more colleagues with whom the decision gets discussed before taking action to ensure the instigator isn't stuck out on a lonely limb. Freedom is there, but so is discipline. Employees feel accountable to each other for results—and they hold each other's collective feet to the fire.

While accountability is driven by culture and individual responsibility, it can't be operationalized without data, without transparency. At Morning Star, everyone sees the data, twice a month, from everyone else and from the company as a whole. Transparency assists employees to make decisions that are in the best interest of the company as a whole, not just their individual business units. Accountability is also woven deeply into the culture by using detailed feedback from a peer review process at both the individual and business unit levels.

There are other, similar examples in Hamel's book, but the same conditions for success seem to be woven through each case. While the details are often different, the broad categorizations of success factors are the same: a dedication to the values of self-management/self-organization, deeply engrained accountability at the individual and organizational unit levels, and comprehensive management support systems. You have to have structure, but whether it develops in a top-down or bottom-up fashion makes a big difference. Self-organization isn't easy; in fact, based on the evidence, it may be harder to carry off than traditional management. Even so, the rewards in terms of performance and organizational health can be substantial.

Effective Collaboration

The agile community promotes the value of collaboration in teams, although "promotes" may be too weak a verb given our fascination with collaboration. While collaboration can have many benefits, in practice we often act as if it is mostly about talking to each other. It's really far more than talking. An effective collaborative team relies on three critical elements: discussion, decision making, and commitment. While healthy discussions are a key piece of collaborative effectiveness, without decision making and commitment, it's just a lot of talking.

Whether you are part of a nine-person iteration planning meeting or one of a pair of developers who are coding, decisions need to be made. Many of them are easy, but some difficult ones can seriously test team cohesion. Why is decision making so difficult? This difficulty can usually be traced to three aspects: focusing on activities and not on outcomes, lack of decision-making processes, and an individual/team paradox.

Several years ago I wrote an article on decision making in software project management. In reviewing six well-known project management books, I found one paragraph on decision making! There were hundreds of pages describing in detail how to "do" stuff, but little guidance on how to make critical decisions. The focus was on all the myriad of activities of project management, except one of the most critical ones: how to make good collaborative decisions—in other words, how to decide on the outcomes of a process. Processes don't deliver results; decisions do.

The lack of emphasis on outcomes, versus activities, extends to developing processes for decision making itself. Few organizations have processes or training around decision making. (Specific processes are outside the scope of this chapter; see *Agile Project Management* [Highsmith, 2010] for one specific decision-making process.) Talking and deciding are two distinct activities, and doing the former well doesn't mean doing the latter well. Part of the reason decision making remains difficult is that it brings out the often hidden paradox between individuals and teams. There is a lingering concern that teams arrive at mediocre decisions and that "smart" individuals make better decisions. The paradox rises in part from Western culture, especially in North America, which seems to value rugged individualism rather than the power of community. This cultural bias gets injected into organizational decision making, creating ambivalence about how decision should be made. Like any paradox, it's not a problem with a solution, but rather an issue that has varying resolutions depending on the timing and context—that is, some decisions may be made by individuals and others by teams depending on the circumstances.

Finally, effective teams make commitments—to each other and to the organization. Commitments within a single team can be informal (in recording—not in the commitment itself), while in large projects with multiple teams there needs to be a more formal interteam commitment mechanism. Team working agreements are an example of making a formal commitment to each other about certain behaviors. Commitments need to be taken seriously and monitored regularly. Everyone misses a commitment at times (because of overcommitting, failure to execute, or just miscommunication), but these misses should be corrected quickly. A series of misses by an individual or a team should be examined in a retrospective.

Part of operating as a self-organizing team-of-teams in a larger project involves commitments between teams. These can be documented using a special type of story card I refer to as an Interteam Commitment Story. This creates a peer-to-peer commitment where one team is the customer and the other is the delivery team. These roles can get interesting, as I've often seen agile teams that make poor customers: they don't want to spend the time performing typical customer activities such as defining requirements and doing acceptance testing—things they expect their own customers to do.

Clearly, then, collaboration is far more than talking. Talking is easy. Reaching decisions can be hard. Creating effective decision-making processes can be hard. Overcoming an individual/team paradox can be hard. Following through on a commitment can be hard. Even so, all of these things are critical to effective collaboration in organizations.

Leadership and Decision Making

A good leader has to be a visionary, a teacher, a motivator, a facilitator, and other things, but he or she must also be a decision maker. The same is true of lead engineers for technical issues. The question then becomes, At what point does a leader's decision making damage self-organization? Of course, when the team loses respect for the leader. But what causes loss of respect? The answer: when the manager begins making unilateral or arbitrary decisions. The more unilateral decisions, the less participation from the team, and the less likely the decisions are to be implemented effectively.

Every team and situation is different, so there isn't a quantitative answer to the question of how many unilateral decisions are too many. However, even though presenting absolute numbers risks misinterpretation, I think the following guidelines may help define appropriate "levels" of leader decision making that will continue to foster self-organization. This rough guide is to make one unilateral decision every month or two, to make three to four decisions per month with team involvement, and then to delegate the hundreds of other decisions to the team. In practice, few good managers make completely unilateral decisions—they usually talk issues over with at least key members of their team. Occasionally, however, there is a need to get things moving by making a unilateral decision. In that same vein, it is appropriate for leaders to make certain decisions with team participation, but if they are making more than three or four of these decisions per month, even with team involvement, they are probably too absorbed in the details.

Another issue related to management decision making is the leader's job of absorbing ambiguity. In fast-moving product development efforts in which key decisions must be made quickly, consensus (unanimous) decision making fails, but even

participatory decision making can get mired down in discussion and debate. Many product development issues, both technical and administrative, may be fuzzy and ambiguous. In these cases, after participation has evolved to a certain point, managers have to be willing to make final decisions: "Well, the information available to us isn't crystal clear, but to move forward with the project, we'll go in this direction." Leaders often have to bring clarity to ambiguous situations—and teams work better because of it. Self-organization does not mean abdication of leader decision making, but rather careful evaluation of when and where each entity needs to make decisions.

Good leaders have earned the credibility to make these decisions. The technical staffs respect the leader's judgment (based on previous actions taken), participate in the analysis and debate process, and willingly accept the decision to move on. The leader has absorbed the ambiguity of the situation, whereas leaving the decision to consensus would have bogged the project down in interminable debate. Good leaders know when to step in and take charge versus when to encourage the team to take charge. They also know when to dig into why team decision making isn't working as it should.

The Usefulness of "Over" in Decision Making

The Agile Manifesto was written in a very deliberate style—for example, "Individuals and interactions over process and tools." The word "over" was carefully chosen and establishes a key agile principle that while many things in our world are too complex for black-and-white answers, we do need to differentiate between what is critical and what is important.

To continue with the example, it's not that process and tools are unimportant—a myriad of tools can increase the productivity of agile teams—but that in the final analysis, people are more important. Think of it this way. If you were a project manager, would you rather have the best tools and processes but mediocre people, or talented people and so-so tools and processes? Obviously, we would pick the latter. Of course, that selection doesn't make tools or process unimportant: how would you like to run a great team whose members had no tools? It's just that sometimes we have to make hard decisions and having a series of "over" value statements can help.

Software development reflects the business world today—complex, uncertain, fast, risky, volatile. These traits dictate that development efforts have to be adaptable, customized, and evolutionary. There isn't a single correct practice or method for every project. Of course, this doesn't mean that anything goes—that there aren't preferences. While we have to be adaptable, we also have to make decisions, and ultimately decisions reflect higher and lower priorities and we need to give people guidelines for their decisions.

What if the Agile Manifesto had been written like this:

> "We believe the following are the most important:
>
> - Individuals and interactions
> - Working software
> - Customer collaboration
> - Responding to change"

While these assertions would have established importance, they would not have been as effective as decision-making guidelines. Even with using the term "over," people still misinterpret the Agile Manifesto as singularly focusing on the first parts of the statements.

In a world of complexity, this use of "over" statements can be beneficial in decision making because such statements do two things: (1) they clearly establish what has highest priority, and (2) they establish that the second part of the statement identifies something important, just not the most important thing. The very fact that "processes and tools" appear in the Agile Manifesto establishes them as important.

Another example of using "over" statements would be in evaluating project performance. What's more important—predictability or adaptability? Writing the statement one way, as "adaptability over predictability," provides us with one guideline, while reversing it provides another guideline. However, by including both, we establish that both are important and need to be considered in our decision making.

One of the most difficult things team members, managers, and executives—everyone, really—do is make decisions. Guidelines, as some criteria for making decisions, are very valuable. But for the most part they can't be black or white—they can't be statements like "In all cases, this is the most important." There needs to be nuance that allows for complexity, uncertainty, and other change-agent factors. Using "over" statements as guidelines is one way to improve your team's and organization's decision making.

RIDING PARADOX

What is an adaptive leader or manager? There are countless answers to this question revolving around the desirable characteristics, mindset, or behaviors—for example, collaborative, light touch, servant, and failure tolerant. One of the critical traits is that of "and" rather than "or" leadership. The most pressing issues to face leaders are usually paradoxical; they appear to have contradictory solutions. Consider, for

example, the paradox of needing predictable delivery while also needing to be flexible and adapt over the life of a project. Agile teams face difficult choices because managers haven't addressed this paradox. They often continue to admonish teams to do both, without really giving them direction about how. Alternatively, they may give lip service to adaptability and focus on delivering to plan—scope, schedule, and cost—just like in waterfall days. Or worse, they may focus on velocity and forget quality.

The ability to ride paradox can be enhanced by integrative thinking:

> Integrative Thinking is the ability to constructively face the tensions of opposing models, and instead of choosing one at the expense of the other, generating a creative resolution of the tension in the form of a new model that contains elements of both models, but is superior to each. (Rotman, n.d.)

Agile teams succeed, in part, because they embrace seeing reality, the reality that "stuff" happens during a project and that the path to success involves adaptation. Ambiguity, risk, and uncertainty are an integral part of innovative projects today. As such, they offer leaders paradoxical situations—situations that require backing away from the direct paradox and figuring out inclusive solutions. Adaptive leaders need to become "riders of Paradox." In Michael Bergt's sculpture "Paradox" (Figure 4.4), the horse seems always to be going in opposite directions at the same time. Furthermore, the leader is exposed, drawn by the traditional norms of many organizations in which it's acceptable to be wrong, but not acceptable to be uncertain.

Figure 4.4 Paradox (by Michael Bergt)

Agile leaders need the courage to view issues from different perspectives, to gather data without undue prejudice, to formulate both/and rather than either/or resolutions. Too few organizations make it past what I've labeled "prescriptive agility," which should be an oxymoron, but unfortunately isn't. These organizations are as rigid about their agile implementations as they were previously about their heavy methodologies! They fail to move beyond rules to understanding. Adaptive leaders need to be riders of paradox, always thinking "how can I do this, AND that" at the same time.

> "Learn the law very well, so you will know how to disobey it properly."
>
> —The Dalai Lama

I'll illustrate with three other examples from software development, issues that have been written about as either/or cases: CMM versus agile, BUFD (big upfront design) versus NUFD (no upfront design), and Scrum versus Kanban. In each case, proponents on either side have set the other up as an enemy to be defeated, rather than considering what is useful in each. The bottom line is that all models are flawed (waterfall, PMI, CMM, Deming, Scrum, XP, Kanban, Lean), but all are also potentially useful. The true adaptive leader—whether an iteration manager, a project manager, a technical lead, a development vice president, or a CIO—attempts to "include" the best from different models. At the Agile 2010 conference, Max Keeler from the Motley Fool discussed using Kanban on maintenance projects and Scrum on larger projects. Scott Ambler from IBM has a wealth of statistics from surveys that show most agile organizations use "just-enough" upfront design. It's easy to be an "or" leader. Pick a side and state your case loudly, over and over again, until the opposition gives up. It's much more difficult to be an "and" leader, balancing between seemingly opposite strategies. However, in our ever-changing and turbulent world, slavishly following the "one right answer" is a recipe for disaster.

Embracing Paradox

> All of us like to think that human affairs are essentially rational. . . . The wealth of experience that fails to support this notion never seems to faze us. . . . That human affairs usually work not rationally but paradoxically. . . . because of the sense of omnipotence that plagues American management, the belief that no event or situation is too complex or too unpredictable to be brought under management control. (Richard Farson, *Management of the Absurd,* 1997)

The ability to "ride paradox," is one of the four management behaviors that define adaptive leadership. We live in a culture of absolutes—think of the current rhetoric of our political parties—but most people recognize that reality imposes a lot of gray. If we think human (or business) affairs are rational, then we attack everything as

a problem to be solved by highlighting the issue, gathering facts, looking for root causes, formulating solutions, and implementing the solution. Once we're done, it's problem solved and on to the next problem.

Astute managers have learned that most serious issues are not really problems, but rather paradoxes that arise again and again. Paradoxes aren't solved once and for all; they require balancing actions again and again. There is even difficulty finding a word for the outcome of a paradox. A "problem" has a solution, but what can we call the outcome of a paradox—a "temporary solution"? The best word seems to be "resolution," which has a dynamic aspect to it that "solution" doesn't. So, *problems have solutions, paradoxes have resolutions.*

Take, for example, the issue of short-term versus long-term focus. There isn't a single answer to this question; instead, a balancing needs to occur from one time frame to another. When a company is in serious financial trouble, working on a 5-year strategic plan probably isn't a good use of management time.

Agile proponents, and opponents, often get hung up on the issue of architecture—to develop architecture upfront or to evolve architecture over time. This isn't a problem; it's a paradox. There isn't a single solution to the question, but rather a series of balanced resolutions that depend on the specific organizational and project or product context. Ultimately, the architecture issue requires a balancing of early skeleton work combined with evolutionary updates. Balancing early versus pursuing evolution makes management more difficult than finding a black-and-white problem solution, but balancing resolutions over time will deliver far better performance.

The ability to differentiate between problems and paradoxes and the further ability to balance paradoxical resolutions time after time is one of the defining characteristics of an adaptive leader. This ability doesn't come easily because discernment and judgment are involved. Paradoxes that agile managers face include the following:

- Accountability versus autonomy
- Hierarchical control versus self-organization
- Predictability versus adaptability
- Efficiency versus responsiveness

None of these is a problem, and none has an easy solution. Any resolution must contain elements of both—delicate balancing acts that change over time. Take the issue of predictability versus adaptability, which crops up on many agile projects. The traditional mandate that "We must be within 5% (or whatever number) of our schedule or cost plan" just doesn't drive teams in the right direction if we want them to learn and adapt over time. Conversely, not giving any predictions to management doesn't

work either. The following factors might influence one's balancing of predictability and adaptability:

- The particular measures used—value, cost, schedule, and so on.
- The level of precision—epics or stories.
- The time frame—iterations, releases, or visionmap (roadmap).
- The relative importance of the measures (it might be value on one project, cost on another).
- Timing: near the end of a project, schedule might dominate, whereas in the beginning of the project, value might.

Living with paradoxes means giving up the notion that we are in control and can dictate (plan) the future. At the same time, we cannot take a totally cavalier "let's wait and see what happens" approach. Living with paradox means planning, but not becoming wedded to that plan. It means sensing when actual events override the plan and responding with the appropriate "resolution." Learning to do this well is a keystone of adaptive leadership.

Balancing Adaptability and Predictability

One of the most vexing issues for executives and managers is balancing the need for both predictability and adaptability in software delivery, or really any delivery. While this might seem like an either/or issue, the solutions are really both/and ones. In presentations I often ask the question, "How many of you have been on agile teams and were asked to be agile, adaptive, and flexible—but also asked to conform to the plan?" Mostly I get snickers and mild laughter at the question because most have experienced it.

Ambiguous, uncertain, and fast-moving environments create paradoxes that cannot be resolved by canned answers. Teams and managers need to understand how to manage these paradoxes by finding ways to do both—to predict and adapt within the same delivery effort. So here are some of my thoughts about riding this paradox.

Time Boxing

One of the key ways that agile teams predict schedule (which is often the most critical attribute for management) is to time-box the effort. While a time box specifies the schedule for a release of the product, it's really more about forcing hard tradeoff decisions than about timing per se. Agilists prefer to adjust scope to meet time deadlines. This approach has proved very effective. Some people refer to it as a release train—as in the train is leaving the station on time and features are either onboard or not. Other managers are uncomfortable with scope adjustments, viewing it as an excuse to underperform—which in some cases has been true. Scope adjustments, as in any complex tradeoff situation, must be visible and defensible.

Confusing Plans and Targets

One big problem, in either traditional or agile projects, is confusing plans and targets—and the level of commitment to each. A target is a goal, usually set by management and usually associated with a date, based on the business situation. The situation could be a regulatory requirement or a manager's assessment of when a new product needs to be launched. It is not an engineering estimate of the project. A plan incorporates the best engineering estimates that can be made at the time (not very good sometimes in the beginning). The target is basically "I need it by June," and the engineering estimate is basically "Given these conditions, we can deliver the product by August." The discrepancy serves as the basis for negotiation—less scope, more people, and so on. Engineers should not be upset when given targets: they are expressions of a valid business need. Managers should not be upset at engineering estimates: they are expressions of a thoughtful analysis of the work required. Plans are the results of negotiating these two perspectives until common ground is reached.

Multiple Levels, Multiple Time Frames

Traditional project teams often plan an entire project in mind-numbing detail (as in thousands of detailed tasks). Agile teams do skeleton planning upfront and then undertake continuous planning and replanning as the project proceeds. With larger and longer projects (which we try to avoid but can't always), the planning should be broken into roadmaps (or visionmaps, as my colleague Ken Collier likes to call them; 6–18 months), releases (3–6 months), and iterations (1–2 weeks, unless you are using Kanban). Each of these levels should be planned at a different level of precision: Capability (40–90 days of effort or equivalent story points), Feature (10–39 days), and Story (2–10 days). Visionmaps should be considered targets and not commitments. Commitments at the Capability level should be general (as in "We will deliver functionality that will meet this Capability"), rather than a commitment to detailed requirements. The shorter the time horizon and the greater the granularity, the greater the commitment level should be.

Short Time Horizons

One way to ease the predictability/adaptability predicament is to shorten the deployment time horizon. While this strategy is product dependent, many organizations can do much better in this regard. And I'm not just talking about the software delivery organization. When we practice continuous delivery, for example, and the cycle time for feature delivery is reduced, the business units must be prepared to deploy, use, support, and promote the new features on a daily or weekly basis, depending on the iteration frequency. Trying to predict 3 months in advance is much easier than predicting (actually much more like guessing) 18 months out. Having short time horizons is actually the best way to solve the predictability/adaptability paradox.

False Predictability Requirements

Finally, there are just those times when requirements for predictability are false, or just ludicrous. I recently had someone say, "We can't go agile because our customers need for us to commit to a plan 18 months in advance." I responded to this remark with my traditional agile consultant question: "So how's that working out for you?" Many times such customer requests are a product of a long dysfunctional relationship with their software delivery organizations. The results of that dysfunction are manifested in weird ways: "We have never delivered according to our plan for the last 5 years, so let's try to predict delivery even further out!" In many situations, solving the dysfunctional relationship problems (which agile delivery often helps to do) can solve the predictability/adaptability problem.

COMPLEXITY AND LEADERSHIP

> Mountaineering at extreme levels is not about skill—many have comparable individual skills. It is not about strength and stamina—although they play an important part. Ultimately, extreme mountaineering is about judgment. It is walking the narrow edge between success and oblivion. (Highsmith, 2000)

There are several important points from this mountaineering analogy that translate into our thinking about adaptive leadership. Mostly it's about judgment.

First, speed is frequently the safest alternative. Customers are often so starved for any results that they become ecstatic about whatever is delivered in 3 to 6 months. By the time 12- to 18-month or longer projects are delivered, relationships are ruined and the product's reception is unenthusiastic at best.

Second, make sure the terrain is where you want to be. Make sure your organization has the skills and abilities to tackle the complexity of the undertaking. Finding yourself crossing a 3000-foot-high, 75-degree ice slope, armed with only crampons and an ice axe, is not the time to realize you should be somewhere else.

Third, the experience needed to hone judgment comes from testing limits in increasingly demanding environments. Higher on the mountain, the critical decision is always between continuing and retreating. Pushing limits is one thing; ignoring risks is another. There is a big difference between taking an informed risk and being reckless. The best mountaineers know when to advance and when to retreat. On one trip they push themselves to incredible limits; the next attempt they abandon the climb early. They understand the environment and its risks, and are able to judge their skills against them.

Last, decisions and actions are the result of complex information and interactions. There are guidelines in the mountains, but no rules. Rules can work in moderate terrain, but team members who know the exceptions to the rules hold the key to success on extreme terrain. Skill and judgment allow the mountaineer to mitigate risk, not eliminate it. Ignoring risks heightens one's dependency on luck, and luck alone is a poor long-term strategy.

Speed, terrain selection, judging risks (margin of error), and understanding the difference between rules and guidelines are all part of traversing dangerous software mountains. Managing in complex environments is no different.

Adaptive leaders deliver enterprise agility. Through both actions and mindset, they have the ability to guide organizations through our world of speed, complexity, uncertainty, and ambiguity. Enterprises of the future that need to respond to these remarkable business conditions must be networked, flat, collaborative, fast, focused, adaptive, and not traditional command–control structures. Growing adaptive leaders to manage this transformation will be key to future success.

Chapter 5

Final Words

"There must be enough chaos in one's life to give birth to dancing stars."

—Friedrich Nietzsche

The agile community has always been full of ambitious, technology-savvy, disruptive thinkers. When you get past pair programming, daily stand-up meetings, and iterations, agility has really been about greatly improving software delivery performance and creating more engaging, exciting working environments. When you get past doing agile (the practices) to being agile (the mindset and behaviors), you find that agile, adaptive, and Lean approaches are more about self-organization, innovation, disruptive thinking, fast learning, constant feedback, interpersonal dynamics, and culture.

Agile software delivery is only the first step. To survive and thrive in today's fragmented, uncertain business world requires adaptive leadership—at all levels of technology and business. Adaptive leadership revolves around three key concepts:

- Envisioning a responsive enterprise
- Delivering a continuous flow of value
- Creating an adaptive, innovative culture

Envisioning a responsive enterprise means building an organization that can manage the constant, unrelenting number of opportunities that flow past your organization every day. Thousands of opportunities must be sifted through, experimented with, and decided upon before settling on those key opportunities—those ambitious missions, or in Nietzsche's words those "dancing stars"—that will carry your organization forward.

And this responsive culture goes past profit; it goes to a purpose beyond the corporate and reaching to the societal—to our ambitious missions for social and economic justice. Engaging people into corralling opportunities takes more than a monetary desire. Engagement comes from purpose, the ultimate ambitious missions that in some way address making the world a better place by furthering social and economic justice.

But identifying a flow of opportunities is only half the battle, and maybe not even the most important half. Execution—bringing products and services to customers, again

and again—requires leaders who embrace technology. Embracing technology means understanding what new technologies can do and what they can't. It means being able to relate technology back to the business purpose—not technology for technology's sake, but technology to achieve your mission. And this point can't be emphasized too much: it's the *continuous flow* of value—being able to deliver time after time, opportunity after opportunity, consistently—that's the key to success in our fragmented world.

Finally, the core of agility, flexibility, or responsiveness—whatever term is your favorite—is culture. My favorite quote related to culture comes from Virginia Postrel:

> How we feel about the evolving future tells us about who we are as individuals and as a civilization: Do we search for stasis—a regulated, engineering world? Or do we embrace dynamism, a world of constant creation, discovery, and competition? Do we crave predictability, or relish surprise? These two poles, stasis and dynamism, increasingly define our political, intellectual, and cultural landscape. The central question of our time is what to do about the future. And that question creates a deep divide. (Postrel, 1998)

Management has been defined as being about the present, leadership as being about the future. Adaptive leadership provides a leadership approach and style that works best to address the dynamism, uncertainty, and possibilities of that future.

Bibliography

Appelo, J. (2011). *Management 3.0: Leading Agile Developers, Developing Agile Leaders.* Upper Saddle River, NJ: Addison-Wesley.

Austin, R. D. (1996). *Measuring and Managing Performance in Organizations.* New York, NY: Dorset House.

Beck, K. (2000). *Extreme Programming Explained: Embrace Change.* Reading, MA: Addison-Wesley.

Bogsnes, B. (2008). *Implementing Beyond Budgeting: Unlocking the Performance Potential.* Hoboken, NJ: John Wiley.

Boyd, J. R. (1995). The Essence of Winning and Losing (a five-slide set by Boyd).

Christensen, C. M. (1997). *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail.* Boston, MA: Harvard Business School Press.

Collins, J. (2001). *Good to Great: Why Some Companies Make the Leap and Others Don't.* New York, NY: Harper Business.

DeMarco, T. A. (1987). *Peopleware.* New York, NY: Dorset House.

Denning, S. (2010). *The Leader's Guide to Radical Management: Reinventing the Workplace for the 21st Century.* San Francisco, CA: Jossey-Bass.

Farson, R. (1997). *Management of the Absurd.* New York, NY: Free Press.

Fulghum, R. (1991). *Uh-Oh: Some Observations from Both Sides of the Refrigerator Door.* New York, NY: Ballantine Books.

Godin, S. (2010). *Linchpin: Are You Indispensable?* New York, NY: Portfolio.

Goldratt, E. M. (1984). *The Goal: Excellence in Manufacturing.* Croton-on-Hudson, NY: North River Press.

Hamel, G. (2009, February). Moonshots for Management. *Harvard Business Review, 87*(2), 91–98.

Hamel, G. (2012). *What Matters Now: How to Win in a World of Relentless Change, Ferocious Competition, and Unstoppable Innovation.* San Francisco: Jossey-Bass.

Hay Group. (2010). *Best Companies for Leadership Study.* Hay Group.

Highsmith, J. (2000). *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems.* New York, NY: Dorset House.

Highsmith, J. (2010). *Agile Project Management: Creative Innovative Products.* Upper Saddle River, NJ: Addison-Wesley.

Hock, D. (1999). *Birth of the Chaordic Age.* San Francisco, CA: Berrett-Koehler.

Horney, N. (2007). Executive Briefing on Agility. Retrieved from Agility Consulting, Inc.: www.agilityconsulting.com

Humble, J., & Farley, D. (2011). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation.* Upper Saddle River, NJ: Addison-Wesley.

IBM. (2010). *Capitalizing on Complexity: Insights from the Global Chief Executive Officer Study.*

IBM. (2012). *Leading through Commections: Insights from the Global Chief Executive Officer Study 2012.*

Isaacson, W. (2011). *Steve Jobs.* New York, NY: Simon & Schuster.

Kanter, R. M. (1983). *The Change Masters.* New York, NY: Simon & Schuster.

Kauffman, S. (1995). *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity.* New York, NY: Oxford University Press.

Keller, S. A. (2011). *Beyond Performance: How Great Organizations Build Ultimate Competitive Advantage.* Hoboken, NJ: John Wiley & Sons.

Kidder, T. (2009). *Mountains Beyond Mountains: The Quest of Dr. Paul Farmer, a Man Who Would Cure the World.* New York, NY: Random House.

Kotter, J. (2012, November). Accelerate! How the Most Innovative Companies Capitalize on Today's Rapid-Fire Strategic Challenges—and Still Make Their Numbers. *Harvard Business Review, 11,* 44–58.

McFarlan, W., & Benko, C. (2003). *Connecting the Dots: Aligning Projects with Objectives in Unpredictable Times.* Boston, MA: Harvard Business School Press.

Michalko, M. (2010, March 27). http://creativethinking.net/WP01_Home.htm.

Moon, Y. (2011). *Different: Escaping the Competitive Herd.* New York, NY: Crown Business.

Moore, G. A. (2011). *Escape Velocity: Free Your Company's Future from the Pull of the Past.* New York, NY: HarperCollins.

Perrin, T. (2007). Global WorkForce Study. Retrieved from Towers Perrin: http://www.towersperrin.com/tp/getwebcachedoc?webc=HRS/USA/2008/200802/GWS_handout_web.pdf

Petzinger, T. J. (1999). *The New Pioneers: The Men and Women Who Are Transforming the Workplace and Marketplace.* New York, NY: Simon & Schuster.

Pink, D. (2009). *Drive: The Surprising Truth about What Motivates Us.* New York, NY: Riverhead.

Pixton, P., Nickolaisen, N., Little, T., & McDonald, K. (2009). *Stand Back and Deliver: Accelerating Business Agility.* Upper Saddle River, NJ: Addison-Wesley.

Postrel, V. (1998). *The Future and Its Enemies.* New York, NY: Touchstone.

Reinertsen, D. G. (2009). *The Principles of Product Development Flow: Second Generation Lean Product Development.* Redondo Beach, CA: Celeritas.

Ries, E. (2011). *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses.* New York, NY: Crown Business.

Ross, J. C. (2010). The IT Unit of the Future. *Center for Information Systems Research*, *XI*(X), 1–3.

Rotman. (n.d.). Integrative Thinking. Retrieved from Rotman Business School: http://www.rotman.utoronto.ca/integrativethinking/definition.htm

Sikes, H. S. (2013, February). Competing in a Digital World: Four Lessons from the Software Industry. Retrieved from https://www.mckinseyquarterly.com/Competing_in_a_digital_world_Four_lessons_from_the_software_industry_3058

Smith, P. A. and Reinertsen, D. G. (1997). *Developing Products in Half the Time: New Rules, New Tools,* 2nd ed. Hoboken, NJ: John Wiley & Sons.

Sull, D. (2009). *The Upside of Turbulence: Seizing Opportunity in an Uncertain World.* New York, NY: Harper Business.

Taylor, W. C. (2011). *Practically Radical: Not-So-Crazy Ways to Transform Your Company, Shake up Your Industry, and Challenge Yourself.* New York City, NY: HarperCollins.

Thomke, S. (2003). *Experimentation Matters: Unlocking the Potential of New Technologies for Innovation.* Cambridge, MA: Harvard Business School Press.

Tushman, C. A. (April 2004). The Ambidextrous Organization. *Harvard Business Review, 82*(4), 74–81.

Ulrich, Dave and Smallwood, N. A. (2006). *How Leaders Build Value: Using People, Organization, and Other Intangibles to Get Bottom-Line Results.* Hoboken, NJ: John Wiley & Sons.

Weinberg, G. (1986). *The Secrets of Consulting: A Guide to Giving and Getting Advice Successfully.* New York, NY: Dorset House.

Weinberg, G., & Smith, S. (2000). *Amplifying Your Effectiveness: Collected Essays (The Satir Change Model).* New York, NY: Dorset House.

Williams, L. (2011). *Disrupt: Think the Unthinkable to Spark Transformation in Your Business.* Upper Saddle River, NJ: Free Press.