METAGILITY

Managing Agile Development for Competitive Advantage

David A. Bishop

J.ROSS PUBLISHING

ISBN-13: 978-1-60427-155-3

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

10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data
Names: Bishop, David A., 1970- author.
Title: Metagility : managing agile development for competitive advantage / David Bishop.
Description: Plantation : J. Ross Publishing, 2018. | Includes bibliographical references and index.
Identifiers: LCCN 2018044684 (print) | LCCN 2018060229 (ebook) | ISBN 9781604278040 (e-book) | ISBN 9781604271553 (hardback)
Subjects: LCSH: Agile software development. | Competition. | BISAC: BUSINESS & ECONOMICS / Project Management. | BUSINESS & ECONOMICS / Management.
Classification: LCC QA76.76.D47 (ebook) | LCC QA76.76.D47 .B573 2018 (print)

| DDC 005.1--dc23
LC record available at https://lccn.loc.gov/2018044684

Direct all inquiries to J. Ross Publishing, Inc., 300 S. Pine Island Rd., Suite 305, Plantation, FL 33324.

Phone: (954) 727-9333
Fax: (561) 892-0700
Web: www.jrosspub.com

CONTENTS

FOREWORD

JUST IN TIME!

Just in Time. In the movies, it's the firemen arriving just in time to save the home; it's the posse arriving just in time to capture the bad guys; it's the hero arriving just in time to rescue the child. In manufacturing, it's the parts arriving on the assembly line just in time to keep the product rolling. In logistics, it's product arriving just in time to go straight onto shop shelves and bypass the stockrooms.

Like many other information systems professors, I am fascinated by *just in time*. Innovations seem to diffuse in a way that is driven by new technology. But it isn't just tech. My research has shown how it is driven by a rising problem: a demand for new solutions that is resolved by the availability of new technology *just in time*. Only then does the technology diffuse. For example, the aging population is triggering a falling rate of labor participation, placing new demands for workforce productivity. Along comes a wonderful new range of robotics, *just in time*.

In digitalization, *just in time* is the arrival of digitalization right at the moment we need it. Just when urban travel becomes too expensive, along comes ride-sharing (e.g., Uber). Just when medical clinics become too crowded, along comes telemedicine (e.g., MDLive). Just when socializing becomes too risky, along comes social networking (e.g., Facebook).

Just when digitalization becomes too big, too fast, and too furious, Dr. Bishop's Metagility arrives. It's *just in time*.

The success of companies everywhere now depends on their software development skills. It doesn't matter whether software development is outsourced or insourced. It doesn't matter whether the company is manufacturing cars or brokering real estate. Digitalization infects both the product and the production process. Oh, and markets not only demand these large-scale digital innovations,

they also demand them right now, *just in time*. It's like a perfect storm forming on the horizon.

So how do companies manage software development projects that can keep pace with this fast-paced, large-scale change unfolding in all walks of human existence? Everywhere, digitalized systems are in development and redevelopment. Everywhere, this development needs to be both really fast and really scalable. In this book, Dr. Bishop shows us how to cope and succeed. His basic idea is simple: harness the inertia. The inertia created by this digitalization is strong; its presence is boundless. If we can make this inertia work for us, instead of against us, we can prevail in profiting from this rushing stream of digitalization.

It takes someone with the background of David Bishop to discover how to do this. He is practiced at this problem. He spent many years managing embedded software development projects in coordination with new product design and manufacturing. He knows the ropes. He has the scars. He also knows how to theorize this problem.

Metagility is as simple as it is brilliant. Use agility not only in software development and manufacturing processes, but also in management processes. Using metagility, management agility can integrate agility into the underlying parallel processes of complex new product development. This integrated agility (which Dr. Bishop calls a hybrid approach) provides a means by which to align difficult-to-coordinate parallel processes, such as iterative software development and large-scale hardware manufacturing. Metagility minds the inertia in each process, channeling each into a vortex that curves slower processes around faster processes until they align. Slow and fast processes meet, without breaking their inertia, *just in time*. It's like a project management wormhole that will look like magic if you haven't read about Bishop's vortex.

Few doctoral students will get through my seminar in evidence-based management without hearing me quote Kurt Lewin, "there is nothing as practical as a good theory." The book you are about to read delivers both aspects of Lewin's Maxim: a theory with practical how-to guidance. Metagility provides a means by which managers can channel the inertia created by digitalization. Smart management of this inertia channel can be used to direct the digitalization into a vortex that can deliver large-scale products, well-coordinated speed, and the quality that those of your customers who are going digital will deserve to get, *just in time*.

Richard Baskerville, Regents' Professor
Georgia State University

PREFACE

In my 25 years of experience as an executive, technical leader, and researcher in technology management, achieving product development deadlines on time and under budget while still meeting market needs is and always has been a perennial struggle for most organizations. This problem is compounded by the fact that the rate of market change and technological innovation have increased exponentially in recent years. The well-known computer scientist and futurist Ray Kurzweil put it best:

> *"We're entering an age of acceleration. The models underlying society at every level, which are largely based on a linear model of change, are going to have to be redefined. Because of the explosive power of exponential growth, the 21st century will be equivalent to 20,000 years of progress at today's rate of progress; organizations have to be able to redefine themselves at a faster and faster pace."*
>
> —Ray Kurzweil in *Perspectives on Business Innovation*

In a rush to match the pace that Kurzweil has described, companies and organizations have sought out solutions to get innovative products to market faster and faster. Some of the first of these were lean manufacturing techniques that were first brought forth as part of the widely acclaimed Toyota Production System in the 1970s. Among other attributes, this system resulted in fewer defects and greater customer satisfaction, which led to significant growth in market shares for Japanese auto companies that were once considered incapable of threatening the Big Three. When the original *Agile Manifesto* was developed in 2001, it was very much an outgrowth of these same lean manufacturing and development ideas. Since then, it has become almost ubiquitous in the software industry. Agile concepts are used in some form or another in virtually all software development performed today. One problem, of course, is just that—agile is largely a set of concepts, not a well-defined process, and as such it has often been interpreted and implemented differently across the world. Despite this,

there's no doubt of its benefits, and there is no mistaking that agile concepts are here to stay.

However, in keeping with Kurzweil's statement, much has changed since 2001. First, there is more software than ever in our society. In fact, it's in virtually everything. Brick-and-mortar businesses are being supplanted by virtual ones, and from our dishwashers to our homes and cars, they each have far more software in them today than they did in 2001.

Second, we live in a world of hyperconnectivity. Social media platforms, smartphones, and cloud storage allow for an almost immediate exchange of ideas and opinions resulting in much higher degrees of customer empowerment. Customers have a huge virtual megaphone today that lets providers know right away if things are going well or badly for their product(s). When customers are happy, sales can soar into the stratosphere seemingly overnight, but when they are not, the results can be devastating.

Third, development is more complex today with a plethora of devices, platforms, libraries, frameworks, and languages available—many of which may be open source. Technology companies today tend to have larger and more diverse product portfolios than in 2001. Back then, Google was nothing more than a search engine and Amazon was a great place to buy books. Now, both behemoths are much more.

Fourth, agile has worked well for software developers, but other stakeholders within most businesses have faced challenges with it, including business analysts, project managers, and executives. Although iterative development of working software increases productivity, such stakeholders still have a great desire for more certainty around budgets, timing, and what the end product will ultimately be. As agile concepts suggest, interactions are certainly of paramount importance over processes or tools, but all too often decisions and conversations go undocumented. Communication is more challenging than ever due to globalization and geographically dispersed teams. The timing to get technology products to market seems to be getting tighter and tighter every day. To many such stakeholders experiencing these frustrations, much of the promised benefits of agile remain elusive with many companies achieving only incremental gains.

Finally, it's not just about software anymore. Technology development has become somewhat less focused on software or websites and much more so on smart devices. Smart cars, home automation, smart meters, and smartphones are creating the *Internet of Things* that has become the basis for some of today's most innovative advances. Development of these devices is more complex in nature because they are more than just software; they blend in hardware and firmware as well.

These issues present an entirely new set of challenges. How can a single product composed of software, firmware, and hardware that were developed on different tracks be delivered in an agile way? How do you find out just how agile you really are, if at all? How can agile concepts be leveraged to maximize the capabilities of your organization? Most important, can agile help your company become number one in its market?

Over the past several years I have researched these questions in earnest. Out of this work, I found that stakeholders can manage both market and process agility within challenging environments, such as embedded systems, via a hybrid agility implementation and product genesis to achieve the desired result. This is referred to as the notion of *agile vorticity*, the point at which market and process agility collides to produce *business momentum* at a specific point of innovation within the *agile business vortex*.

To put that in plain English, agile has indeed changed since its 2001 inception, and it is being adapted to maximize innovation and time to market in today's most challenging and complex product development. *Metagility* provides a unique playbook and system of measures based on our most successful case studies that can put your organization into hyperdrive and help your company become number one in its market.

ABOUT THE AUTHOR

Dr. David A. Bishop is a technology consultant, enterprise architect, researcher, and instructor with over 25 years of experience working for clients such as AT&T, BellSouth, EDS, Delta Airlines, Toshiba, and government agencies. In addition to this experience, David holds an ABET accredited Bachelor of Computer Engineering degree from the Georgia Institute of Technology, an MBA from DeVry University with a concentration in IT management, and a Doctorate in Business Administration from Georgia State University. David is CEO and Founder of Agile Worx, LLC (http://www.agile-worx.com), a software development firm that provides program and project management tools based on his research. He is a member and committee chair for the International Electrotechnical Commission (IEC) based in Geneva, Switzerland, a member of ANSI, and a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE) and the Association for Computing Machinery. David is also founding chairman for the Atlanta IEEE Technology and Engineering Management Society. Dr. Bishop is the creator of the agile vortex theory, which is the result of years of intensive practitioner research in some of the world's most successful organizations in their respective industries. Its focus was to study how these organizations managed both agile and waterfall techniques to outperform their competition in industries with very high technological change, turbulent markets, and innovation.

David resides in Woodstock, Georgia, U.S.A.

This book has free material available for download from the Web Added Value™ resource center at *www.jrosspub.com*

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

Downloads for *Metagility: Managing Agile Development for Competitive Advantage* include:

1. An agile artifact template: Provides a detailed format for creating requirements and user stories with guidance on how to fill in the blanks.
2. A user story workflow infographic: Provides users with a detailed workflow of how user stories move through the development process. It can be used as a training tool and poster.
3. A requirements workflow infographic: Provides a workflow specific to requirements artifacts and could be used in the same way as the user story infographic.
4. An agile metrics calculator: This spreadsheet provides important metrics-calculating tools for project managers or other team leaders.

Part 1

The Need for Metagility

Establishing a New Direction for Your Organization

1

AGILE METHODS TODAY: WHAT HAS CHANGED AND WHY

INTRODUCTION

Agile methodologies have become a popular and widely respected method for managing software development. Since the inception of the *Agile Manifesto*, agile development methodologies have superseded waterfall methodology in many, if not most, software development organizations. Despite its apparent success, however, many organizations have struggled with the adoption and implementation of agile and exactly what level of adoption provides the most agility. Agility is commonly held in the literature to be constructed of elements external to a company or project but, in fact, may be composed of both external and internal elements. The exact relationship of the *adoption* of agile development techniques to a company or organization's *true agility* remains unclear. Crucially, there are no measurements available for each of these factors in an embedded software development context nor is there a method to determine the impact of one against the other. A major reason for the lack of measurements is due to the somewhat amorphous definition of agile itself. In academic literature, the concept is still relatively young and poorly defined. In practice, organizations have largely opted for a hybrid approach to agile, mixing its concepts and method with existing stage-gate or waterfall methodologies. This has made the management of agile even more complex. These issues beg the following questions:

- How do organizations orchestrate and assess agility?
- How is agile best managed in difficult or complex situations?
- How can some of the commonly known shortcomings of agile be mitigated?

- Most important, given the apparent efficiency and performance gains of agile, how can it be leveraged to maximize organizational performance and product success?

Metagility answers these questions by examining multiple dimensions of the issue (agile definition, history, adoption, implementations, hybrid solutions, project/program management) to determine how agility processes are orchestrated (*meta-agility*) in a number of companies *that were successful in leveraging and maximizing agility to become number one in their respective market*(*s*). Throughout this book, we highlight one of these case studies in particular. A study of an embedded systems development organization was conducted with twelve managers from hardware, firmware, and software development groups within the company. These managers included product, project, people, and technology managers with direct responsibility for managing agility processes. An innovative form of research, referred to as grounded theory, was utilized to perform data analysis and determine a *central phenomenon* or theory at the crux of the agile question. This research revealed the central phenomenon to be *agile vorticity*—the point at which maximum agility is achieved. This point is reached through the management (meta-agility) of two main subcategories: process and market agility. The relationships of these categories are illustrated with a thought experiment of a whirlpool vortex. Included is an empirical account of how agility can be managed and tweaked for optimum results. In the chapters that follow, we examine market and process agility to determine the success factors for achieving agile vorticity and becoming number one in your market!

We begin by establishing a historical perspective of agility—its evolution and its current state as compared to when it was first developed.

THE *AGILE MANIFESTO* TODAY

The *Agile Manifesto* as it was originally written in 2001 is a set of four now very well-known tenets:

- Individuals and interactions *over* processes and tools
- Working software *over* comprehensive documentation
- Customer collaboration *over* contract negotiation
- Responding to change *over* following a plan

The idea being that while all elements in the previous statements have value, the ones on the left side are valued even more. In this section, we will explore each of these four tenets and how today's environment is influencing them to change.

Individuals and Interactions over Processes and Tools

In 2001, people worked more independently as waterfall processes tended to limit interactions and focus on well-defined goals. Agilists hoped to break this lack of interaction by emphasizing teamwork, but much of this was predicated on colocated teams. As Table 1.1 illustrates, teams today and the communication channels they use are more complex. Many development organizations are distributed all over the world and use a variety of communication methods and tools. With that in mind, tools themselves have become much more sophisticated and user friendly. Many such tools are easily customizable and have agile concepts built in.

Table 1.1 Globalization and technology: illustrates how the agile tenet of *individuals and interactions* has changed since 2001

2001		
Individuals and interactions	OVER	Processes and tools
Today		
Individuals and interactions	PLUS	Processes and tools
Globalization has led to more complex teams and interactions		Today's tools have matured: they facilitate interactions and drive processes
Results of interactions must be documented		Agile is a process

Waterfall was extremely *heavy* with a plethora of burdensome steps and operating procedures designed for CYA (cover your ass!) rather than focusing on creating a great product. On the other hand, some aspects of agile, such as team collaboration, have proven challenging to implement at scale. Although interactions are certainly very important, the lack of process around documenting the outcomes and decisions of these interactions can result in a great deal of *churn*. Churn in this context means burning resources and losing time due to repetitive meetings and rework.

Many organizations have found that today's tools can facilitate conversations and can be used to drive, not inhibit, development. Additionally, they have also discovered that implementing agile itself is a process, even if it is much smaller in scope than waterfall was. The right combination of process and tools can be decisive in the success of your business.

Working Software over Comprehensive Documentation

Older waterfall processes often resulted in cumbersome and exhaustive *word documents* which included requirements documentation, design documents, standard operating procedures, maintenance documents, support documents, and much more. Emphasis was placed on writing many of these documents up front before attempting to build anything. This *comprehensive documentation* tended to inhibit response to change and resulted in a slowdown of innovation and productivity. Documentation is still needed for today's development, but it takes a different form. It is no longer a series of heavy *word documents*, rather it comes in the form of features, epics, user stories, and other artifacts that may be documented within collaboration tools or other applications. Table 1.2 illustrates these differences. As mentioned previously, outcomes of key interactions need to be documented, just not in a process heavy or cumbersome document, but within a lean and easily managed set of tools.

Table 1.2 Products and documentation redefined: illustrates the changing tenets of *working software* and *documentation* today as compared to earlier this century

2001		
Working software	OVER	Comprehensive documentation
Today		
Working product	WITH	Documentation
Products are more complex and are not just software anymore		Artifacts such as epics, user stories, and decision outcomes are documented with tools as opposed to long-form comprehensive documents

The need for documentation is even more critical with today's complex products such as smart devices or Internet of Things (IoT) applications that are created using multiple teams, development tracks, and sub-products.

Customer Collaboration over Contract Negotiation

Contracts are a fact of business life. They aren't going away. Regulatory requirements, idiosyncrasies of specific industries, or the nature of certain products require them. In today's world, contracts and collaboration are not mutually exclusive; in fact, they are anything but. When we discuss contract negotiation, that negotiation within itself is a form of collaboration with the customer. The difference is that this collaboration is not necessarily a one-time thing where

all items are discussed and documented up front in a contract, rather it is an ongoing process of communication and alignment.

Collaboration has become more than just talking directly to the customer. Often, the customer may not be able to articulate exactly what they need, and it may require a variety of stakeholders on both the customer side and internal to the development organization to glean succinct requirements and break those down into increments that can be developed into a working product. Table 1.3 illustrates this change.

Table 1.3 Collaboration *is* negotiation: negotiating with the customer, including the contract, is a form of customer collaboration

2001		
Customer collaboration	OVER	Contract negotiation
Today		
Customer collaboration	INCLUDES	Contracts and negotiation
Collaboration happens faster via a variety of internal and external stakeholders		Collaboration and negotiation are one and the same

Responding to Change over Following a Plan

In 2001, *plans* were synonymous with *rigid*. A plan was often considered to be a representation of the final work. Making changes midstream once the plan was finalized was discouraged, and if changes were implemented, it often resulted in a painful approval process and reevaluation of delivery timelines, costs, and performance expectations. A great deal of work was performed up front to ensure that the plan, and the requirements it entailed, were as complete as possible so that developers knew exactly what they needed to do down to the finest detail. Different interpretations of the plan, unanticipated engineering challenges, and inability to accommodate changing customer needs often resulted in failure. Today, markets are simply moving too fast to adhere to this kind of rigid approach. At the same time, there must be some kind of plan to ensure that the company is getting the business value out of the development work that not only makes the customer happy but also satisfies their strategic goals. This means that the definition of a plan has changed. As Table 1.4 illustrates, no longer is it a long, exhaustive document that is full of requirements and design specifications, rather it is more of a vision for the organization. Change is inevitable, but every change must be evaluated against the values and needs of the business.

Table 1.4 Vision versus plan: accelerated market pressures have caused rigid, short-term project plans to be replaced by more flexible, long-term visions

2001		
Responding to change	OVER	Following a plan
Today		
Responding to change	WHILE	Following a vision
Change must always be evaluated for business value and adherence to strategic goals		Plans have been replaced by visions

Now that we understand how the *Agile Manifesto* has changed from a practitioner standpoint, let's take a look at what the business research says about the topic.

AGILE MANAGEMENT TODAY: A RESEARCH PERSPECTIVE

The management of agile methodologies can be a challenge to organizations on many levels. The first issue is the concept of agile itself. The current research literature on agility is sparse, particularly in relation to information systems development (ISD). This is largely due to the fact that agile is a relatively new concept in an ISD context and is therefore not entirely solidified. It is, in essence, a set of concepts or ideas, not necessarily a methodology or framework of its own. The second challenge is the management of agility. Most agile-based frameworks available today are operational in nature, focusing on project management indicators such as velocity, release frequency, sprint completion, and so forth (Highsmith, 2010). Still, other methods focus entirely on the outcome of agile adoption in relation to environmental turbulence (Yauch, 2011). This type of study, however, does not allow for the evaluation of the adoption of agile principles, such as people over processes and tools, against quality and customer responsiveness. Giachetti holds that the assessment and management of agile should not only consider performance, but agile characteristics that have been assimilated into the organization (Giachetti et al., 2003). Although there have been methods describing agile adoption, none of these has related the level of adoption to agility outcomes. As a result, these existing methods do not completely address all dimensions of agility. These challenges are compounded by the fact that recent research has revealed that most organizations have not adopted agile in its entirety, but instead have assimilated a variety of

agile concepts and methods into existing traditional methods. Such *hybrid agile* implementations have only added to the complexity of agility management.

Orchestration of agility within any organization is often poorly defined, due to its fluid, quickly changing, and somewhat amorphous concepts, processes, and methodologies. Hybrid implementations, environment specific assessments, and varying degrees of market turbulence are just a few dimensions of agility that should be considered within a management context. In short, the management of agile methods requires their own brand of agility. Our goal from a research perspective was to examine our case studies against the various dimensions of this agility to determine how agile processes are typically managed and orchestrated.

WHY ARE AGILE METHODS AND AGILITY CONCEPTS SO CRITICAL TO BUSINESS SUCCESS NOW?

The short answer to that question is that it is becoming increasingly difficult to stay ahead. Technology today is changing at an ever-accelerating rate. In his essay "The Law of Accelerating Returns," futurist Ray Kurzweil said that this rate of acceleration is occurring exponentially and could eventually result in "technological change so rapid and profound it represents a rupture in the fabric of human history." We see evidence of this change in our everyday lives, not just in technology. Companies must work harder and harder to keep up, tweaking ever more agility out of their processes and teams.

Hyper-Accelerated Markets

Many case studies in our research experienced *hyper-accelerated markets*, which created some very unique conditions. One of these conditions is what is referred to as an *agile vortex* or *whirlpool*. Agile vortices are created when extreme levels of market pressure are torqued by high innovation or quickly evolving technology. This provided an excellent laboratory under which it was possible to observe how a company achieved *optimum* performance, or essentially *agile nirvana* in today's most challenging markets. This phenomenon is referred to as reaching the point of *agile vorticity*, which will be explained in a later chapter.

How these hyper-accelerated markets take shape can be illustrated by close examination of one of our cases studies. In this case, the technology developers were in the smart grid market, developing smart meters for power utilities that wanted to upgrade their grids to the latest *green technologies*. These are the same kind of meters that you may find on the side of your house or business; the

difference is they are automated, more accurate, and in most cases communicate wirelessly without the need to send out technicians to read them. Not only do smart meters save money and protect the environment, they provide greater security as well. The subject of our case study faced several challenges which could be summarized in the following ways.

Extreme Innovation

1. *Carrier and networking technology was rapidly changing*: From low power radio frequency (RF) signals to cellular to power line carrier, there are a host of communication methods that could be used to transport meter data from the field to the utility. Adopting the best technology could make or break a company. Our case study had invested deeply into RF mesh networking technology and was constantly faced with new threats.

2. *Meter capabilities dramatically increasing*: Originally fairly dumb devices, electric, gas, and water meters were acquiring the ability to manage smart devices throughout the home, communicate pricing to customers, and gather detailed event and power data. In many ways, the growing capabilities of the available technologies were outpacing the company's ability to get them tested and implemented—not to mention the fact that competitors were in the same race. The same thing was happening, and in fact still is happening, with other IoT solutions. Our case study was rushing to create the most intelligent smart meters of all!

Extreme Market Pressure

The customers for the subject of this case study were power utilities. In many cases, governments in many parts of the world provided these utilities with incentives to switch to smart meters. At the same time, these smart meters had a long life span of 10 to 20 years, thus, once a utility selected a vendor, that customer would be out of the market for quite a long time. There were several start-up smart metering vendors and only a fixed number of *big utilities*. This created a *land grab* situation where each vendor was rushing to capture as many big utilities as they could. Whoever managed to garner the largest chunk of early market shares would dominate the market for a LONG time. This combination of incentives, long-term contracts, and limited customer base resulted in extremely high market pressure.

Combined, these extremes of market pressure and innovation created a hyper-accelerated market. One in which innovation and market pressure are so great that they result in a virtual hurricane for vendors to navigate.

What Is the Solution?

Transitioning to, adopting, and implementing agile methodologies in a software organization is a costly and time-consuming proposition. Companies and organizations need to know where they stand in terms of adoption/assimilation and its impact to the external agility of the business. They also need a framework to manage their performance so that strategic adjustments can be made. We set out to conduct studies that would provide greater understanding with regard to how agility can be managed and the impact of assimilation and adoption on these management processes. As we will see in a review of the latest research literature, most embedded systems environments, such as the one mentioned previously, with larger, more mature organizations, have taken a hybridized approach to agile adoption by combining agile with existing stage-gate methodologies. The orchestration of the processes required to support agile principles in such an environment and the way those principles are affected can reveal new insights into agility in new and different contexts. Such contexts include:

- Large teams,
- Geographically distributed teams,
- Hybrid agility evolution, and
- Embedded systems (environments which include the synchronization of firmware, software, and hardware development).

APPROACH FOR BUILDING A NEW DIRECTION

The experience of practitioners such as project managers, developers, testers, program managers, engineering managers, and executives is of the utmost importance. The challenge is gleaning out this knowledge and experience, analyzing it, and producing new knowledge and insights that can be applied in useful ways by other practitioners repeatedly in the same situations. All too often, consultants or other practitioners try to create methodologies or frameworks based on trial and error or completely in a vacuum, relying on their own limited experiences to determine the best path forward. In many such cases, such experts will create an approach that simply matches what they think the client wants, thereby producing a work that makes the customer happy in the short run but produces lackluster long-term results. The best way to approach business problems today is through what we call engaged scholarship. In a nutshell, engaged scholarship is the concept of applying scientifically valid research techniques to solve business problems and develop innovative business solutions.

This technique produces work that is not only scientifically valid, but also applicable to a variety of situations and contexts in a repeatable fashion. It does not rely solely on the limited experiences of a few, or on the specific opinions or methods of a small number of consultants. Instead, it takes a holistic view of what is going on in a particular market and produces a solution that, in short, is of much higher quality than approaches that do not make use of business research. In this work, a scientific approach has been taken to ensure that the results have the highest integrity and validity, therefore providing the highest value possible to practitioners in the field. In this section, the approach to developing the theoretical aspects of this work is described.

This research consists of an interpretive case study using a grounded theory analysis. Grounded theory is an innovative form of qualitative analysis that is particularly well suited to studying information systems, technological development, and innovation in a business context.

The grounded theory approach makes use of the Straussian brand of grounded theory outlined in Strauss and Corbin's *Basics of Qualitative Research: Grounded Theory Procedures and Techniques* (Strauss and Corbin, 1990). Such studies can provide theories arising from the research effort itself, and therefore do not generally employ theoretical frameworks as a lens to examine a problem. However, such studies can use other research to inform the study at hand (Strauss and Corbin, 1990). This case study uses the four basic agile principles mentioned previously, as stated in the *Agile Manifesto,* for this purpose.

This is a participative study that includes perspectives on agile from a variety of stakeholders within the various case study subjects. It examines the normative questions dealing with the management, design, and evaluation of agile assimilation that is specific to embedded systems development. These traits are characteristic of an engaged scholarship effort using design and evaluation research (Van de Ven, 2007). Contributions of the study include:

- An empirical account of how agility principles and methods are orchestrated in embedded software development, and hence, the orchestration of hybrid agility,
- A method or framework for orchestrating agility over time in context, while constantly adapting and fine-tuning it, and
- Greater insights into the drivers of agile process innovation.

Any good study begins with a review of the current research literature in the subject area at hand. In the following sections, important works in the fields of software, technology, and agile development are brought forth for the reader's benefit. This is part of the process of ensuring the validity of what is being done here. In addition, it is also an interesting review of the history of agile and the

forces that brought it about. Having a greater understanding of what brought agile development to its current state will enhance our ability to develop future solutions. We begin with an exploration of engineering and software traditions, followed by agility, embedded systems, and the current state of research on each.

THE EVOLUTION OF AGILITY: ENGINEERING AND SOFTWARE TRADITIONS

In the past, there has been much debate as to whether software development is considered to be science or engineering. In the literature, most researchers believe it to be the latter. This perception is due in large part to the difference between the tools that scientists and engineers use. While a scientist may use experimentation; the engineer develops prototypes or demonstrations. In his study on the traditions of computing, Tedre stated that *computer science is an empirical science not based on traditional scientific experimentation* (Tedre and Sutinen, 2008). Over time, software development became more regarded as an engineering discipline as the size and scale of computing projects required larger teams (Tedre and Sutinen, 2008). As a result, the guiding mantra for software engineers and computer scientists is more often considered to be *demo or die*, as opposed to the traditional *publish or perish* employed by so many traditional researchers. These precepts have had a significant influence on the management of large software development efforts. This section seeks to explore the evolution of software development management and how this evolution has been influenced by engineering, science, and innovation.

Although larger teams and more complex development projects have led to software development becoming more of an engineering discipline, this has also led to a need for a more efficient means of management. Early software engineering managers and developers were faced with challenges in design, development, communication, and production. Arguably, the earliest form of software development management is traditionally referred to as the software development life cycle or *waterfall* method. It is also often referred to as a stage-gate method due to its rigid process orientation and the use of *gates* to pass from one phase to another. The creation of this method is most often attributed to Winston Royce—dating back to 1970 when he documented the process he used to develop software packages for spacecraft mission planning (Royce, 1970).

Although not originally intended to be a comprehensive roadmap, Royce's work proliferated throughout other government agencies and became widely adopted within the software development industry. According to Christiane Floyd in her 1992 publication of "Software Development and Reality

Construction," this software development tradition is based on the following precepts (Floyd, 1992):

- Software engineering is produced based on a series of fixed requirements.
- These requirements are provided via an analysis process that is performed before design begins.
- Developers are only responsible for producing a solution that meets these specific requirements.
- Software development is independent of individuals. Developers are considered to be interchangeable resources.
- Communication should be managed and regulated through fixed interfaces.

Floyd argues that while this existing methodology has brought about impressive advances in programming methods and allows for a greater understanding of software development before coding begins, it does not support the subsequent emergence of insights into functionality, implementation, and usability (Floyd, 1992). The need to provide such insights gave rise to a greater usage of the engineering concept of prototyping.

The concept of prototyping, or even rapid prototyping, is not new. Rather, it is a tradition that has been a part of electrical and mechanical engineering for many years. One reason for its success is that it allows for more teleological requirement definitions based on outputs and constraints (Orr, 2004). Back in 1985, many authors believed that prototyping would replace traditional lifecycle methods of systems development (Janson and Smith, 1985). However, a study from this time period does not recommend that. Instead, it performed a comparison of prototyping based on existing engineering practices against proposals to use the same technique in information systems design. This comparison suggested that prototyping should be integrated within existing waterfall processes in order to:

- Verify user requirements,
- Verify design specifications,
- Aid in selecting the best design,
- Assist with various stages of testing and development, and
- Gain approval for new product concepts (Janson and Smith, 1985).

This need for prototyping in software has given rise to iterative software development in which regular demonstrations of incremental components are critical to developing an entire product line.

Manufacturing concepts have also had a significant impact on software development. The concept of lean or *just-in-time* manufacturing became prominent

in the 1970s as part of the highly regarded Toyota Production System. Lean development is based on removing any aspect of a process that does not add customer value. At Toyota factories, inventory was kept to a minimum by only manufacturing the necessary products, at the necessary time, in the necessary quantities. In addition, the system included a *respect-for-human* system that allowed workers the latitude to display their capabilities by improving their own work processes (Sugimori et al., 1977). Lean techniques improved the flow of information and materials across the business, focused on market pressure created by the customer, and required an organizational commitment toward continuous improvement. Although lean manufacturing was driven in large part by Japan's need to compete with Europe and America with fewer resources, the concepts became widely adopted in the automotive industry all over the world.

The production control of the Toyota system was referred to as the *Kanban* system, which consisted of a series of order cards instead of computers to manage production (Sugimori et al., 1977). Kanban is yet another outgrowth of the Toyota Production System which has made its way into software development management. Using Kanban, software development organizations have been found to reduce the amount of work in progress. Such lean concepts of value and waste elimination have been found to provide *target and route* for continuous software development improvement, particularly with agile development (Wang, Conboy, and Cawley, 2012). Organizations that employ Kanban techniques have moved away from the time-boxed iteration to more of a *continuous flow*. This has been shown to be especially true in organizations with mature adoption of agile development techniques (Wang, Conboy, and Cawley, 2012).

Agility

Agility originated—in a manufacturing context—primarily as an output of lean or flexible manufacturing (Mathiassen and Pries-Heje, 2006; Kidd, 1995; Dove, 2002). It concerns the economy of scope, as opposed to scale (Dove, 2002). It has been defined as the ability to manage and apply knowledge effectively, to adapt to change (Arteta and Giachetti, 2004; Dove, 2002), and it has been summarized as the capability to quickly respond to market requirements (Ramesh and Devadasan, 2007). The concept of agility was first introduced in a report from the Iacocca Institute (Nagel and Dove, 1991). Other notable research articles include:

- A Hewlett-Packard (HP) agility assessment expands on agility as comprising three factors of speed, range, and ease to assess an organization's ability to respond to change (HP, 2005).

- Agility was defined by Goldman, Nagel, and Preiss (1995) as the ability to prosper in a competitive environment characterized by constant and unpredictable change. This concept was further broken down into four dimensions of agility, which include:
  - Enriching the customer,
  - Cooperating to increase competitiveness,
  - Organizing to control change and uncertainty, and
  - Leveraging the impact of people and information.

Haeckel expands on these dimensions in a different way by enumerating the organizational characteristics that are required to achieve agility as described in Goldman's dimensions.

The speed with which an organization can respond to customer requests, market dynamics, and emerging technical change is seen as a key element. This includes:

- Time to sense relevant events,
- Time to interpret what is happening and assess the consequences for the organization,
- Time to explore options and decide which actions to take, and
- Time to implement the appropriate responses (Haeckel, 1999).

Organizational capabilities, both tangible and intangible, that provide the basis for conducting business and creating change are also considered to be a prerequisite to achieve agility. These include people, technology, processes, and knowledge (Haeckel, 1999).

Adaptability is also essential. How well organizations respond to changing demands, threats, or opportunities depends on their ability to learn and to use flexible processes and products that can be reconfigured without extensive additional costs (Haeckel, 1999; Dove, 2002; Mathiassen and Pries-Heje, 2006).

Agility has perhaps been best described as a solution for maintaining competitive advantage during times of uncertainty and turbulence in the business environment (Sharifi and Zhang, 2001). Further research into the literature reveals that agility is more than a method or an organizational capability. Rather, it is a business philosophy (Highsmith, 2010). As agile concepts champion people over processes, focus should be on the organizations and the individuals that comprise them. Agile *minds* should be quick, resourceful, and adaptable in character. Agile *organizations* should respond quickly, be resourceful, and be able to adapt to their environment (Mathiassen and Pries-Heje, 2006).

Indeed, organizations are complex adaptive systems. Such systems have been defined as being comprised of decentralized independent individuals interacting in self-organizing ways, guided by a set of simple, generative rules, to create

innovative emergent results (Highsmith and Cockburn, 2001). Highsmith and Cockburn emphasize creativity over written rules as the way to manage complex software development problems and diverse situations (Highsmith and Cockburn, 2001). It is this style of management from which *organizational* agility is said to have arisen. Additionally, organizations do not think objectively about software development agility (Sheffield and Lemétayer, 2013). Adoptions of agile practices are typically subject to the organizational structure and context. In one study, it was found that developmental organizations—those which focus on adaptation and creativity—were much more conducive to the adoption of agile practices than a hierarchical culture that focused more on command and control (Iivari and Iivari, 2011).

Although the concept of agility has been available in manufacturing for some time, it was not adapted to ISD until the advent of the *Agile Manifesto*. Information systems have brought about a new context and application for agility. The ability of an ISD to support agility has been defined as the continual readiness to rapidly or inherently create change, embrace change, and learn from such change while contributing to perceived customer value. Such value is often characterized as additions to economy, quality, and simplicity. This is accomplished via the collective components of the ISD and its relationships with its environment (Baskerville, Pries-Heje, and Madsen, 2011).

Based on the literature that has been discussed, it can be noted that agile is somewhat conceptually weak from a research point of view due to the number and variation of definitions. In practice, however, it is considered to be defined well enough for *practitioners*—particularly with respect to how they use and combine agile with plan-driven methods (Baskerville, Pries-Heje, and Madsen, 2011).

Agile Software Development Methodologies

Agile is an iterative software development methodology based on self-organizing and cross-functional teams. Although mentioned at the beginning of the chapter, it is reiterated here with reference as interpreted by the research. It is based on the following key concepts derived from the highly popularized *Agile Manifesto* (Alliance, 2001; Vinekar, Slinkman, and Nerur, 2006) that argues:

- Individuals and their interactions are more important than processes and tools
- Working software is more important than documentation
- Customer collaboration is more important than contract negotiation
- Responding to change is more important than following a plan

Using Ward's method, Dingsøyr et al. (2012) identify the seminal works in agile research and many of the key underlying themes. Most of the early research focused on understanding agile concepts. Other key topics included adoption or adaptation, reconciliation between agile and plan-driven methods, and evaluation of adoption issues in environments not conducive to agile. Prior to our study, more research is needed to better define what the core of agile is and its role in architecture and knowledge management (Dingsøyr et al., 2012). Additional research was also found to be needed with respect to examining agile across various contexts such as different projects and organizations.

One such context that demands study includes methods for ISD. An ISD method can be defined as one that *encompasses the complete range of practices involved in the process of designing, building, implementing, and maintaining an information system; how these activities are accomplished and managed; the sequence and frequency of these activities; as well as the values and goals of all of the above* (Conboy, 2009). Such a method is not a set of rules, but an ideal in the sense that it is not expected to be followed literally.

Conboy finds that an agile ISD method should meet the criteria of:

- Flexibility—the ability to create change (proactively, reactively, or inherently)
- Ability to embrace change in a timely manner through its internal components and relationships with its environment
- Leanness—the ability to contribute to perceived customer value through economy, quality, and simplicity from the *customer's perspective*

Agility is the combination of flexibility and leanness with continual readiness. Qumer adds speed, learning, and responsiveness to these criteria (Qumer and Henderson-Sellers, 2008). One problem with current agile method thinking is that some practices are now commonly referred to as agile even though the connection to the concept may be tenuous at best, and even if this link is clear, it may be too simplistic to be considered agile in every context or circumstance (Conboy, 2009). Conboy states that the following steps should be taken to evaluate such practices based on the aforementioned criteria:

1. Evaluate whether certain practices or procedures are agile with respect to long-term sustainability and implementation.
2. Examine the *behaviors* and *outcomes* that contribute to agility.

This idea could be extended by developing assessments to evaluate performance outcomes (Conboy, 2009). Applying such assessments across methods, method variants, organizations, and projects could not only reveal interesting insights, but improve orchestration of agile processes. As we shall see in a later chapter,

there are several *agile frameworks* in the market today that would not pass a critical agile assessment.

The integration of agility concepts into ISD has created a number of variations on a theme. These include adaptive systems development, dynamic systems development, test-driven development, and feature-driven development to name a few. By far the most popular of these methods used in the industry today are XP and Scrum (Baskerville, Pries-Heje, and Madsen, 2011). A more recent addition to this list of commonly used methodologies is Kanban.

Scrum, XP, and Kanban

Scrum is more of an agile management methodology with a greater focus on projects, while XP is more of an engineering philosophy concentrated on code management and quality (Wang, Conboy, and Cawley, 2012). Both methods can be, and often are, used in tandem. Weaknesses with XP have been cited with medium- to large-sized projects because of inadequate testing, architectural planning, and documentation—all of which are often required with large complex systems in which the cost of change can be high (Qureshi, 2012). Studies have shown that some shortcomings such as defect rates can be mitigated by extending XP to include more analysis and architectural design—characteristics which look similar to waterfall-based methods (Qureshi, 2012). For the purposes of this study, we are primarily focused on the management aspects of agile and, therefore, the focus will be on Scrum. Another reason for this emphasis is that Scrum is commonly used in hybrid implementations of agile methods. This is mostly because Scrum acts as a wrapper around existing development methodologies and can be used with virtually any existing method (Schatz and Abdelshafi, 2005).

Despite its success, Scrum has been found to present challenges with resource allocation. This has led to a noticeable shift in the industry from agile to lean software development practices (Wang, Conboy, and Cawley, 2012). Kanban is one such process, and it is a less-structured method than Scrum, which focuses on minimizing the amount of work in progress. Instead of using time-boxed iterations, Kanban employs more of a flow by allocating time and resources as they are needed. Software development organizations that have challenges with work estimation and interruptions have been shown to display improvements over lead time and defect rates by using Kanban over the time-boxed method of Scrum (Sjøberg, Johnsen, and Solberg, 2012). A significant area for further research is operational guidance on mapping such lean and agile processes to their current roles and a roadmap for implementing them (Wang, Conboy, and Cawley, 2012).

Successful implementation of an agile methodology has been found to rely heavily on the establishment of many cultural and procedural changes within an organization. The first of these is building a continuous feedback loop to allow for constant replanning (Vidgen, 2009). Shared responsibility by empowering Scrum team members to manage day-to-day work is also important (Vidgen, 2009). A key enabler for self-managed teams is fostering high communication and collaboration on a daily basis. Spontaneous interactions should be supported by structured interconnected practices such as Scrum meetings and pair programming (Vidgen, 2009). A willingness to adapt the process to the development context is crucial, and the development iterations themselves should work toward a sustainable rhythm (Vidgen, 2009). Agile implementation is not limited to the development organization. Product management should also integrate agile methods into their work. This can be accomplished by establishing sprints that alternate with the development teams. Implementing agile in product management as well as in development provides for structured detailing of complex requirements, early collaboration, and disciplined backlog administration (Vlaanderen et al., 2011).

For requirements prioritization, it was found that a mix of agile and plan-based methods proved to outperform either agile or plan-based methods alone in which volatility is not very high or low (Port and Bui, 2009). Since volatility is rarely at the extreme and often unknown, it was inferred that mixed strategies should be the most widely used. However, it should be noted that very turbulent markets or *gold rush* situations could accelerate the volatility or *pull* rapidly. The adoption of such mixed practices has been found to allow for change, driven by close customer interaction, continuous requirements gathering, and frequent iterative delivery (Vidgen, 2009).

Although agile has been noted to increase productivity, foster shared learning, and create job satisfaction among developers (Vinekar, Slinkman, and Nerur, 2006), it may not be the best choice for all environments. The literature as well as practitioner experience shows that successful adoption of agile in its purest form may depend on the following factors:

- Size of the project and team,
- Consequences of failure or criticality of the project,
- Volatility of the environment,
- Skill level of the development team(s), and
- Company culture (Vinekar, Slinkman, and Nerur, 2006).

One example of such an environment—embedded systems development—is impacted by these factors on many levels. Firmware and hardware development teams tend to be much smaller than their software counterparts with a much higher degree of specialization. While there are many software developers in the

organization with skills that are easily transferable from one project to another (such as C#, Java, or .NET programming), their firmware counterparts do not share the same level of transferability. Firmware professionals often have very focused knowledge of the embedded systems stacks for home area networking, RF network communication, or metering metrology that inhibits them from being easily interchangeable. Embedded systems are often *mission critical* systems with high consequences for failure. As a result, organizations developing such systems tend to be less comfortable with the higher rates of change that often come with the iterative and potentially chaotic agile development than their software counterparts.

HOW HAVE MOST COMPANIES DEALT WITH THESE ISSUES?

Simply put, technology products are developed using a *process*, and companies usually deal with the issues mentioned in the previous section by implementing a process that is tailored to their organizational needs and market. As we stated earlier in the chapter, processes are here to stay, and agile development is a process. As we have shown, such processes have changed over the years, beginning with the stage-gate process that came from the space program in the 1960s, to the famed Toyota production system of the late '70s that touted *lean manufacturing* techniques and the concept of *continuous improvement*. At the dawn of the new millennium, lean manufacturing concepts began making their way into software development in the form of *agile methods*. Today, virtually every vendor of technology products—from cell phones to software—utilizes some form of agile to create and bring their products to market. Some of the challenges encountered with technology development processes include:

1. *Productizing innovation*: Technology is constantly changing and vendors are always seeking out ways to tweak their process so that they can get new innovative technologies to market quicker.
2. *Responding to market pressure*: Customers want the latest technologies and there are always competitors out there that are working just as fast as you are to get those new technologies to market. Often, the first out of the gate will have the best chance at capturing the largest market share.
3. *Overcoming organizational resistance*: Vendors must make the necessary changes within their organizations in order for the *improved process* to work. This is often easier said than done.
4. *Maintaining quality*: Customers don't want an inferior product. Period. Quality is defined as giving the customer what they want. Steve Jobs was successful because he had an intuitive sense of what customers wanted—and provided it.

Achieving *Agile Vorticity*

So what have the companies in our best case studies done in this situation? The short answer is that they made specific changes in their process, which allowed them to become #1 in their market. Through a series of organizational and product management changes, they managed themselves into a *sweet spot* which we refer to as *agile vorticity*. As we shall elaborate on in a later chapter, agile vorticity is defined as the point at which a company has achieved maximum agility in its market.

This is important because hyper-accelerated markets are becoming increasingly common and the method of responding to them may become the operational standard for technology developers. Just as our own weather is slowly becoming more turbulent with greater occurrences of hurricanes and tornados, so are markets becoming more volatile.

In the next chapter, we take a deeper look at what the current body of research says about the performance of agile methodologies since its 2001 inception.