"If you want to understand how to lead a Continuous Delivery or DevOps transformation in your company, there's no better book than this. Concise, practical, and based on hard-won executive experience, this book is essential reading for every IT executive." —Jez Humble, VP, Chef

Gary Gruver and Tommy Mouser

LEADING THE TRANSFORMATION

Applying Agile and DevOps Principles at Scale

foreword by Gene Kim

IT Revolution

LEADING THE TRANSFORMATION

Applying Agile and DevOps Principles at Scale

Gary Gruver and Tommy Mouser

Foreword by Gene Kim

IT Revolution
Portland, OR

ISBN 978-1-942788-01-0

IT Revolution
Portland, OR
info@itrevolution.net

For quantity purchases by corporations, associations, and others, please contact the publisher at orders@itrevolution.net.

Cover, illustrations, and interior design by
Brian David Smith and Mammoth Collective.

CHAPTER 1
UNDERSTANDING THE TRANSFORMATION

There are large organizations in the industry using leading-edge techniques such as Agile and DevOps to develop software faster and more efficiently than anyone ever thought possible. These are typically companies that learned how to architect and develop software well while they were still relatively small. They then grew large quickly because of these breakthrough capabilities. Think Google, Amazon, and Facebook.

Currently, however, the majority of software is not developed by leading-edge groups like these, but by more traditional organizations using less efficient approaches. This book is written to help leaders of these traditional organizations understand how to successfully transform their development and delivery processes.

Improving the effectiveness of software development in traditional organizations is essential because software is a key way businesses now compete across a broad range of industries. Mechanical engineers that designed and built cars led the automobile industry. Then, through no fault of their own, they found that computers had infiltrated their product and become a larger part of the value they provide their customers. Now, instead of the salesman showing off the car engine, they start with a screen for the entertainment and control system—all based on software. Financial institutions that used to depend on traders working the floor and brokers forging customer relationships are finding that software for managing trades and interacting with their customers is helping them stay competitive. Retail has gone from building, stocking, and managing stores to creating software that provides a common customer experience across stores, websites, and mobile devices and that manages inventory more efficiently across all these channels.

No industry is immune from the far-reaching changes based on the increasing influence of software. Jeff Immelt, CEO of General Electric, warns, "In an industrial company, avoid [gaining mastery of] software at your peril. We are paranoid because a software company could someday disintermediate GE. I'm going back to school on big data and software."

While it is clear that software is becoming a more and more important aspect of how these companies need to compete, most large, traditional organizations are struggling to deliver. They can't respond to changes in the marketplace fast enough, and the businesses are getting frustrated. These companies are typically struggling with lots of hard-to-change, tightly coupled legacy software that requires them to coordinate development, qualification, and deployment efforts across hundreds to thousands of engineers, making frequent deliveries impossible. The deliveries they do provide require lots of brute-force manual effort that is frustrating and burning out their teams.

The net result is that most large, traditional organizations are finding it more and more difficult to compete in the marketplace and deliver the software innovations that their businesses require. Their current software delivery approaches are constraining their businesses and limiting their ability to compete.

Because their current approaches don't work, many larger organizations are looking to leverage the successes that smaller businesses have seen using Agile methodologies. They bring in Agile coaches and start forming Agile teams to apply Agile principles at the team level. The problem with this approach is that in small organizations, a couple of small Agile teams can organize to support the business. In large, traditional organizations, however, most of the time individual teams can't independently deliver value to the customer because it requires integrating work across hundreds of developers and addressing all the inefficiencies of coordinating this work. These are issues that the individual teams can't and won't solve on their own. This is why the executives need to lead the transformation. They are uniquely positioned to lead the all-important cultural changes and muster the resources to make the necessary organization-wide technical changes.

In this book we, the authors, will provide a fundamentally different approach for transforming the software development processes in large, traditional organizations by addressing the organization-wide issues that you, the executives, are uniquely positioned to handle. While most Agile implementations start with a focus on applying Agile principles at the team level, the approach presented in this book focuses on applying the basic principles of Agile and DevOps across the organization. It is based on what we, as executives leading complex transitions in large, traditional organizations, have found to be most effective for delivering solid business results.

Many specifics referenced in this book are leveraged from a case study of transformation at HP, detailed in *A Practical Approach to Large-Scale Agile Development*, by Gary Gruver, Mike Young, and Pat Fulghum.

This case study includes the following dramatic results:

- Development costs reduced from ~$100M to ~$55M
- 140% increase in the number of products being supported
- Increased capacity for innovation from 5% to 40%

The organization at HP achieved these results through applying DevOps and Agile principles at scale. Our focus was on applying the principles at the executive staff level, and we left the teams with as much flexibility in operational choices as possible. There were some groups that applied all the team-level Agile principles and some that chose to operate with more traditional methods.

What we found in the end is that there were not dramatic differences in the teams' productivity based on the methods they used. There were, though, dramatic improvements in the overall productivity of the entire organization. This lead to our conclusion that *how teams come together to deliver value* in large organizations is the first-order effect, while *how individual teams work* was a second-order effect. Therefore, this book will primarily focus on how to transform the way the teams come together to provide value to the business by integrating all their changes early and often in an operation-like environment. This is one of the most important steps in improving the effectiveness of large organizations, because it forces resolving conflicts between teams early before too much time and effort is wasted on code that won't work together in production. Then, when that part of the transformation is complete, the organization will have the right framework in place to continue improving and fine-tuning how the individual teams work with more traditional Agile methods at the team level.

Executives need to understand that applying Agile and DevOps principles at scale both differs significantly from typical Agile implementations and provides quicker time to value. To help executives understand why, they need to understand the challenges that large organizations experience with traditional approaches. In chapter 2 we will dissect the Waterfall Method, look at the Agile principles that answer Waterfall's shortcomings, and then uncover key challenges that result from using a more traditional Agile approach in the enterprise.

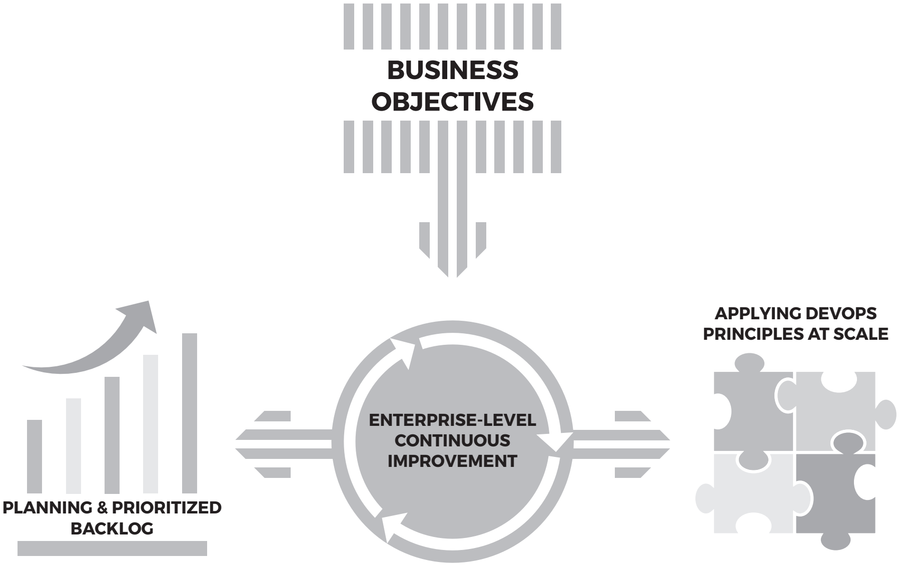
The first step executives need to understand about our approach is that it is paramount to begin with business objectives. You should never "do Agile or DevOps" just so you can say you did them. A large-scale transformation is too much work and turmoil just to be able to say you are "doing Agile." We believe that the key reason executives would be willing to take on this much change is that their current development processes are failing to meet the overarching needs of the business. Executives are in the best position to understand those failings and the needs of the business, so they are best suited to clarify the objectives of the transformation. In chapter 3, we will go into how executives begin to lead the transformation, using these objectives to communicate the vision, prioritize improvements, and show progress.

Once the business objectives have clarified the long-term goals of the transformation, executives then will use an enterprise-level continuous improvement process to engage the organization throughout the journey. Because it is so hard to measure process improvements with software, executives can't just manage the change by metrics like they would other parts of their business. They are going to have to engage with the organization to get a more qualitative understanding of what is working and what needs fixing next. This transformation can't be top-down, just like it can't be bottom-up.

The continuous improvement process is designed to engage the broader organization in setting objectives the team feels are important and achievable. Additionally, since a transformation of this size can take years and is going to be such a discovery process, it is designed to capture and respond to what everyone is learning along the way. The executives will use a combination of the business objectives and the continuous improvement process to lead the transformation and prioritize improvements based on what will provide the biggest benefit to the business. We will cover the continuous improvement process in more detail in chapter 4, including setting short-term objectives for each iteration, focusing on what everyone is learning, and identifying what is and isn't working to determine priorities for the next iteration.

In this book we will use the term *enterprise-level* to describe an organization with software development efforts that require 100 or more engineers to coordinate the development, qualification, and release of their code. It does not refer to a coordinated effort across an organization the size of HP, because that would just be too complex. The plan for transforming the organization should be kept as small as possible to reduce complexity. But if different applications in the enterprise have to be qualified together to ensure they work in production, then they should be included as part of the same enterprise-level transformation.

FIGURE 1


BUSINESS OBJECTIVES
PLANNING & PRIORITIZED BACKLOG
ENTERPRISE-LEVEL CONTINUOUS IMPROVEMENT
APPLYING DEVOPS PRINCIPLES AT SCALE

Once the business objectives and continuous improvement process are in place, executives can start changing development processes by applying Agile and DevOps principles at scale. This will require two big changes: applying Agile principles to the planning process and using DevOps to address the basic Agile principle of being able to economically release smaller batches of changes on a more frequent basis.

Executives need to understand that managing software and the planning process in the same way that they manage everything else in their organization is not the most effective approach. Software has a few characteristics that are different enough from everything else that it makes sense to take a different approach. First, each new software project is new and unique, so there is a higher degree of uncertainty in the planning. Second, the ability of organizations to predict the effectiveness of software changes is so poor that literally 50% of all software is never used or does not meet its business objectives.[1] Third, unlike any other asset in a business, if software is developed correctly it is much more flexible and cheaper to change in response to shifts in the market. If the planning process doesn't take these differences into account, executives are likely to make the classic mistake of locking in their most flexible asset to deliver features that will never be used or won't ever meet the business intent. Additionally, if executives don't design the planning process correctly, it can end up using a lot of the organization's capacity without providing much value. In chapter 5, we will cover how to design processes to minimize investments in planning and requirement breakdown but still support the critical business decisions by breaking the planning process into different time horizons and locking in capacity over time.

The DevOps approach of integrating working code across the organization in an operation-like environment is one of the biggest challenges for large, traditional organizations, but it provides the most significant improvements in aligning the work across teams. It also provides the real-time feedback engineers need to become better developers. For this to work well, the continuous deployment pipeline needs to be designed to quickly and efficiently localize issues in large, complex systems and organizations. It requires a large amount of test automation, so it is important that the test automation framework is designed to quickly localize issues and can easily evolve with the application as it changes over time. This is a big challenge for most organizations, so the executives need to make sure to start with achievable goals and then improve stability over time as the organization's capabilities improve. Teams can't and won't drive this level of change, so the executives need to understand these concepts in enough detail to lead the transformation and ensure their teams are on the right track. Therefore, we spend a lot of time on applying DevOps principles at scale in chapters 6–11.

Transforming development and delivery processes in a large, traditional organization requires a lot of technical changes that will require some work, but by far the biggest challenges are with changing the culture and how people work on a day-to-day basis.

What do these cultural shifts look like? Developers create a stable trunk in a production-like environment as job #1. Development and Operation teams use common tools and environments to align them on a common objective. The

[1] Ronny Kohavi et al, "Online Experiments at Microsoft," Microsoft Research website, accessed April 1, 2015, http://research.microsoft.com/en-us/projects/thinkweek/expthinkweek2009public.pdf.

entire organization agrees that the definition of done at the release branch means that the feature is signed off, defects-free, and the test automation is ready in terms of test coverage and passing rates. The organization embraces the unique characteristics of software and designs a planning process that takes advantage of software's flexibility. These are big changes that will take time, but without the executives driving these cultural shifts, the technical investments will be of limited value.

From business objectives and continuous improvement to planning and DevOps, *Leading the Transformation* takes you through the step-by-step process of how to apply Agile and DevOps principles at scale. It is an innovative approach that promises markedly better business results with less up-front investment and organizational turmoil.

CHAPTER 2
CHALLENGES WITH SCALING AGILE TEAMS

Traditional implementations that focus on scaling small Agile teams across the organization are very different from applying Agile and DevOps principles at scale. Executives play a key role in communicating the advantages of the latter approach and in explaining how it differs from what is typically done in the industry. This chapter outlines the basic Agile principles for executives and highlights the limitations of the typical approach of scaling small teams across the organization. This information is vital to executives looking to avoid the struggles of traditional implementations and to capitalize on the business benefits of a successful transformation.

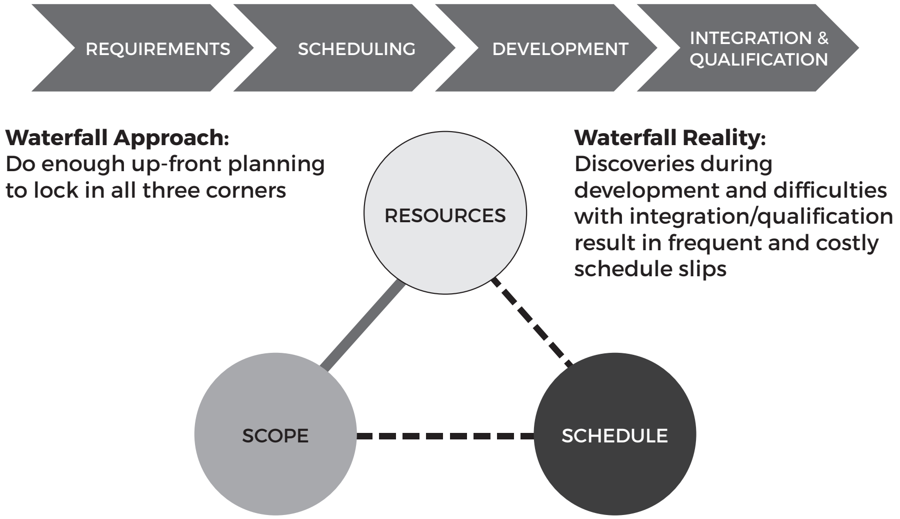
Waterfall Method vs. Agile

As many executives know, the Waterfall Method of development leverages project management principles used for managing many types of projects. It starts by gathering requirements and then planning all the work. Development begins after the planning, and then software is integrated for the final qualification and release. The goal of this approach is to structure the program such that you can determine the schedule, scope, and resources up front.

Large, complex software development projects, however, are fundamentally different than other types of projects, and traditional project management approaches are not well equipped to deal with these differences. Software development is such a discovery process that many of the assumptions made in the planning stage quickly become obsolete during development. Additionally, integration and qualification tends to uncover major issues late in the process, which results in frequent and costly schedule slips and/or working the teams to death.

The first step in leading the transformation is understanding that Agile principles are a response to the shortcomings of using traditional Waterfall project management approaches for software. They were proposed as a framework to address these unique software development challenges.

FIGURE 2: WATERFALL DEVELOPMENT MODEL


REQUIREMENTS
SCHEDULING
DEVELOPMENT
INTEGRATION & QUALIFICATION
Waterfall Approach:
Do enough up-front planning to lock in all three corners
RESOURCES
Waterfall Reality:
Discoveries during development and difficulties with integration/qualification result in frequent and costly schedule slips
SCOPE
SCHEDULE

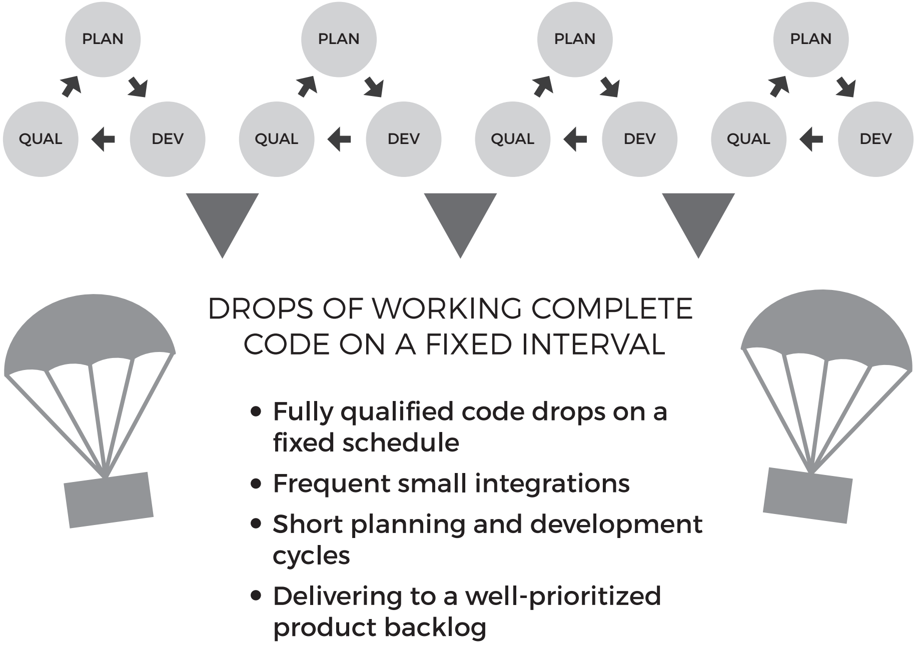
Instead of long phases of requirements, planning, development, and qualification, there are much smaller iterations where complete features are integrated and qualified on a regular basis. Additionally, the entire code base is kept stable so that the code can be released at the end of each iteration, if required. This fixes the schedule and resources while letting the scope absorb the program uncertainty. The features are all worked on in priority order, with the most valuable features being developed first. Agile practitioners have numerous examples where after delivering less than 50% of the original must-have features, the customer is happy with the product and no longer requesting more features.

Contrast this with the Waterfall Method, where there are no regular code drops. The qualification and integration process would not have started until all the must-have features were complete, taking much longer to deliver any value and creating features that may not have been necessary. While there are many other benefits to Agile, this highlights the key breakthrough for the business and as such is imperative for executives to understand when contemplating leading a large-scale Agile transformation.

Change Management Capacity

Transitioning to Agile is a very big effort for a large organization. There are technical and process changes required. Frequently, organizations focus on the

FIGURE 3: AGILE DEVELOPMENT MODEL



DROPS OF WORKING COMPLETE CODE ON A FIXED INTERVAL

- **Fully qualified code drops on a fixed schedule**
- **Frequent small integrations**
- **Short planning and development cycles**
- **Delivering to a well-prioritized product backlog**

technical solutions. While required, they represent a smaller portion of the effort. Most of the real challenges are in organizational change management and shifting the culture. Executives need to understand that the capacity of the organization to absorb change is the biggest constraint to rolling out these improvements. This means that the organizational change management capacity is the most precious resource, and it should be actively managed and used sparingly.

The approach to rolling out an enterprise-level Agile transition should focus on the business breakthroughs the Agile principles are intended to achieve while taking into consideration the capacity of an organization to change. This is where executives can add knowledge and expertise.

The Limitations of Traditional Agile Implementation: An Executive Perspective

What follows is an example of a large (1,000-developer) organization that tries to enable small Agile teams in the organization. The example is not real, but it is a snapshot of what is being done in the industry today.

The first step is to select a few pilot teams with eight to ten members who will start being "Agile." These teams will gain valuable experience and create some best practices. Once these teams have demonstrated the advantages of Agile, they create a plan for how it should be done for this organization. The plan will need to scale across the organization, so in the end there are going to be ~100 Agile teams. Agile coaches will be hired to get a few teams going, and then within a year, all the teams should be up and running. Throughout the year, each coach can probably ramp up about five teams; thus, this rollout could require in the range of 20 coaches. With a cost of $150/hour, this adds up to over $2M/year. When forming Agile teams, it is important to understand that you want teams that can own the whole feature end to end, so you need to pick members from the different component teams to form the prototype teams. This works fine with just a few teams, but when we start rolling this out organization-wide, we are going to need a complete reorganization. Once everyone is in teams creating effective workspaces for collaboration, moving everyone around will probably take another $1–2M.

The next step is making sure the teams have the right tool for documenting and tracking their Agile stories, which will probably run $1M or more. All of a sudden, we are up to over $5M for the transition. We better make sure we talk to the executives who can commit that level of resources. This will require a ROI justification to the CFO. So now we are committed to a big investment and a big return to the top-level executives. At this point, we have an implementation that engages and includes the engineers and the top-level executives. The management team in the middle, however, does not have a clear role except to provide their support and stay out of the way.

There are a number of problems with this whole approach. You are now $4–5M into a transition, and you still don't have a plan for having always-releasable code for the enterprise or an enterprise backlog. Teams may have a clear local definition of "done" in their dedicated environments and team backlogs, but at the enterprise level you have not changed the process for releasing code. Also, this approach has driven lots of organizational change that may be met with some resistance. We started with taking managers out of the process because they are a big part of the problem and don't understand how to coach Agile teams. Can you see why and how they might undermine this transformation? Next, we have a complete reorganization, which always tends to be a cause for concern and creates resistance to change. Add to that moves designed to get teams together in collaborative workspaces. You can see how this approach is going create a lot of turmoil and change while not fundamentally changing the frequency of providing code to the customers for feedback.

The other big challenge is getting the teams to buy in and support how they are going to approach the details of their day-to-day work. The first prototype teams are going to be successful because they had a lot of input in defining the team-level processes and have ownership of its success. The problem most organizations have is that once they have determined the "right" way for their teams to do Agile, they feel the next step is teaching everyone one else to do it that way. This approach of telling the teams how to do their day-to-day work feels like it is contrary to the good management practices that we learned early in our careers.

In most cases, for any problem there are at least three different approaches that will all achieve the same solution. On the one hand, if we were really smart managers, we could pick the best approach and tell the team how to do it. If, on the other hand, we let the team pick how to meet the objective, they are more likely to make their idea more successful than they would have made our idea. If we told them how to do it and it failed, it was clear that we didn't have a very good idea. If they got to pick the *how*, then they were much more likely to do whatever it took to make their idea successful. Just being managers or being part of the prototype team did not mean we were any more likely to pick the best idea.

Therefore, as leaders we feel it is important, wherever possible, to provide the framework with the objectives and let the team have as much design flexibility in defining how the work will get done. It provides them with more interesting work, and they take more ownership of the results. In addition, when the situation changes, those doing the work are likely to sense it and adapt more quickly than an executive would.

Summary

What we hope executives walk away with after reading this example is that most Agile implementations struggle to provide expected business results because they focus on rolling out Agile teams the "right" way instead of applying Agile principles at scale. This approach creates a lot of change management challenges in an organization without fundamentally addressing the basic Agile business principles of an enterprise backlog and always-releasable code. We believe our approach offers answers to a lot of these struggles. Having the executives and managers leading the transformation by setting the business objectives and running the continuous improvement process engages them in the transformation. Focusing on improving the organization-wide planning and delivery processes provides clarity on the business breakthroughs the basic Agile principles were intended to provide. Providing a framework for prioritizing and integrating work across the teams provides the basic processes for improving the effectiveness of large organizations while providing the teams with as much flexibility as possible in defining how they work on a day-to-day basis. What follows is a detailed account of how each step of the process builds on the next. What you end up with is a concrete plan to apply Agile principles at scale to help executives lead the transformation in their own businesses.

To order a copy of *Leading the Transformation: Applying Agile and DevOps Principles at Scale,* please visit www.itrevolution.com.

For quantity purchases by corporations, associations, and others, please contact the publisher at orders@itrevolution.net.