The Shortest Distance Between You and Your New Product

HOW INNOVATORS USE RAPID LEARNING CYCLES TO GET THEIR BEST IDEAS TO MARKET FASTER

Katherine Radeka

2nd Edition

Chesapeake Research Press
Camas, Washington, USA

Copyright © 2015, 2017 by Katherine Radeka. All rights reserved.
First edition 2015. Second edition 2017.

Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

ISBN: 978-0-9795321-6-0

All trademarks are trademarks of their respective owners. Rather than put a trademark symbol after every occurrence, we use names in an editorial fashion only, and to the benefit of the trademark owner, with no intention of infringement of the trademark. Where such designations appear in this book, they have been printed with initial caps.

Chesapeake Research Press print and e-books are available at special quantity discounts for use in corporate training programs or pilot projects. To contact a representative, please email us at bulksales@chesapeakeresearchpress.com.

TERMS OF USE

This is a copyrighted work.

Chesapeake Research Press and its licensors reserve all rights in and to the work. Use of this work is subject to these terms. Except as permitted under the Copyright Act of 1976 and the right to store and retrieve one copy of the work, you may not decompile, disassemble, reverse engineer, reproduce, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish or sublicense the work or any part of it without Chesapeake Research Press's prior consent.

You may use the work for your own noncommercial and personal use; any other use of the work is strictly prohibited. Your right to use the work may be terminated if you fail to comply with these terms.

THE WORK IS PROVIDED "AS IS." CHESAPEAKE RESEARCH PRESS AND ITS LICENSORS MAKE NO GUARANTEES OR WARRANTIES AS TO THE ACCURACY, ADEQUACY OR COMPLETENESS OF, OR RESULTS TO BE OBTAINED FROM USING THE WORK, INCLUDING ANY INFORMATION THAT CAN BE ACCESSED THROUGH THE WORK VIA HYPERLINK OR OTHERWISE, AND EXPRESSLY DECLAIM ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PROBLEM.

Chesapeake Research Press and its licensors do not warrant or guarantee that the functions contained in the work will meet your requirements or that its operation will be uninterrupted or error-free. Neither Chesapeake Research Press nor its licensors shall be liable to you or to anyone else for any inaccuracy, error or omission, regardless of cause, in the work or for any damages resulting therefrom. Chesapeake Research Press has no responsibility for the content of any information accessed through the work. Under no circumstances shall Chesapeake Research Press and/or its licensors be liable for any indirect, incidental, special, punitive, consequential or similar damages that result from the use of or inability to use the work, even if any of them has been advised of the possibility of such damages. This limitation of liability shall apply to any claim or cause whatsoever, whether such claim or cause arises in contract, tort or otherwise.

CHAPTER ONE

Why Rapid Learning Cycles?

In most places, the work to turn an idea into a product consists of one long, slow learning cycle. It's as if we go from San Francisco to Los Angeles by way of New York.

I've been a developer of one kind or another since my first job out of high school. I loved seeing my companies' products—the ones I worked on and my clients' products—get out into the world.

I have bittersweet memories of all-night debugging sessions, last-minute product changes and launch celebrations where we were all too tired to party. I've watched product ideas bounce up and down the priority list. I've seen my projects suffer from last-minute travel restrictions or budget cuts. I've been on the receiving end of phone calls from executives demanding to know why a product would miss its launch date.

Most of the time, the pain was worth the joy of seeing the product on the shelf. But it takes a heavy toll on people. I dedicated the last twelve years of my life to getting the fun and excitement back into product development—to minimize the pain and maximize the joy.

In the last six years, I have worked with world-leading companies to develop a much better way to do product development that is more joyful

and less frustrating. Together, we have figured out how to eliminate those long, slow learning cycles.

Long, Slow Learning Cycles Get in the Way

In most product development programs, we try to run as fast as possible, but our efforts slow everything down.

Our assumptions about development, our eagerness to bring our new idea to life and established Product Development Processes drive us to make decisions early in development. The only information we have to go on when making these decisions is our past experience and our vision for the product. Later, when things don't work out the way we expected, we have to revisit those decisions. In the worst case, we don't learn until after the product has been released and customers complain—or refuse to buy the product at all. We spend too much time on the wrong ideas, while our best ideas suffer from poor execution.

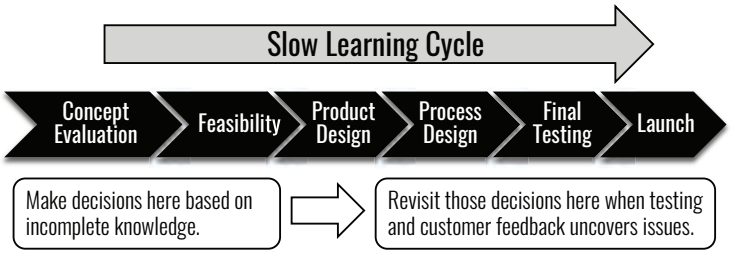

Figure 1.1: Most Product Development Is One Long Slow Learning Cycle

These long, slow learning cycles directly affect the success of our products. When we make decisions too early, we end up in redesign loopbacks to fix problems that crop up, or we release the product without fixing them and suffer the consequences. Sometimes groups try to eliminate the redesign work by locking down decisions even earlier—but that doesn't

work. It just makes the learning cycles longer, because even less information is available to inform the initial decisions.

What if you went on a hike to the top of Dog Mountain but realized after fifteen minutes that you left your car unlocked, and you had to return to the trailhead? Then, an hour later, you found that you left your camera at the creek ten minutes back? Then chose the wrong path at a confusing trailhead?

All of these require us to loop back to an earlier place in the hike. These loopbacks slow our progress, and could keep us from reaching the top at all if we want to finish by nightfall. When we do reach the top, we may be too tired to enjoy it.

Loopbacks plague product developers working in both startup orga-nizations and long-established companies. In a startup, the passion of the product vision, the push to earn revenue before the money runs out and investors lose patience, and the natural optimism it takes to be entrepre-neurs combine to build pressure to move quickly without taking any time for investigation. Teams just dive in and start building stuff using what-ever tools are at hand, and whatever people are willing to take the risk.

In an established company, the product development process can be so risk-averse that great ideas have to jump through a lot of hoops to get funded. This forces decisions that the team isn't ready to make, in order to build a business case that will support the product's investment. Once an idea gets in the portfolio, it becomes difficult to kill —even if it's clear that the product will not achieve its goals. Finally, companies tend to stuff their portfolios with too many projects, so that every project team is starved for resources and driven to take shortcuts.

In any company, the pressure is to make decisions fast and early, defer-ring learning until later, if at all. Intuitively, this makes sense: the best way to learn how to build a product is to build the product.

The Pressure to Build the Product Quickly Slows the Product Down

If we had perfect knowledge of the work to be done to get from here to there, it would make sense to dive in and build. We could make decisions early and then hold people accountable for executing them flawlessly. But that's not how innovation works. If we already know how to do something, it's not innovation.

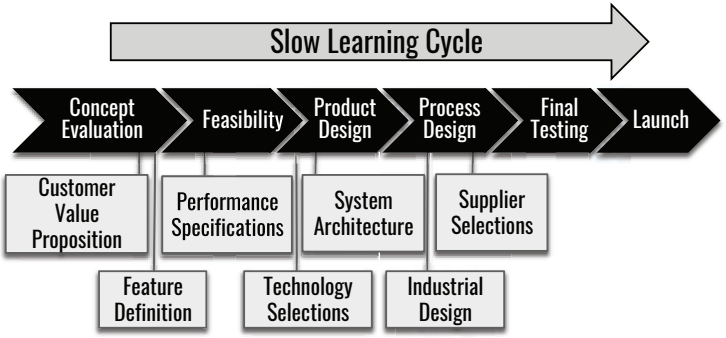

Figure 1.2: A Few of the Sources of Long, Slow Learning Cycles

I once asked a group of product development managers to draw their Product Development Process (PDP) for me. It looked like a long, winding path with uncrossable rivers, blind canyons and unexpected cliffs. The development teams were running as fast as they could toward their goal, but managers (the people making the drawing) were constantly throwing rocks in their way. Meanwhile, the team spent a lot of time building bridges over the widest spots in the river, setting up rappel lines to get down those cliff walls and backtracking out of blind canyons.

In the physical world, the journey with the shortest travel time is rarely a straight line. You can hack your way through the wilderness by following a compass, but that wastes a lot of energy. To find the best route, you need to understand the lay of the land: the natural mountain passes, the animal trails that naturally follow the terrain and the spots in the river

that are easiest to cross. It takes a little extra time and patience, but it saves so much more than it costs.

The Drive to Make Quick Decisions Wastes Time

When decisions come early and learning comes late, you've headed off on a compass direction without understanding the terrain. A lot of unnecessary obstacles get in the way between you and your new product:

- **It takes a long time to move from a good concept to a profitable product that meets customer expectations.** The group gets stuck in a long series of Build–Test–Fix loops. Unanticipated problems that don't have easy fixes tend to pop up during the late phases of development. Launch dates slip, and market windows get missed. Investors get impatient with a team that's stuck in "almost done." Lead customers, who may have helped you design the product, get nervous when they can't meet their own internal milestones.
- **It takes a long time to learn whether or not the idea is fundamentally flawed, and therefore should have been stopped sooner.** The company wastes money on ideas that should have been killed a lot earlier, and the innovators spend too much of their lives working on unsuccessful products that they could have spent investigating new product opportunities.
- **The end product is disappointing.** The execution is not nearly as smooth or well designed as the innovators envisioned. The product may be slower than expected, unreliable, difficult to scale to sufficient volume or clunky in a way that even early customers won't accept. This erodes profitability with low yields, scrap and warranty returns, and erodes customer confidence. The problems

may even require expensive and embarrassing recalls that kill all the profits from the product.

- **The product costs more, from idea through obsolescence.** The drive to "make it work" leads to solutions that add cost to the product at the end, and those costs stay with the product throughout its life. It may be difficult to maintain, difficult to service and even difficult to recycle at the end.
- **Business partners and customers lose confidence in the development team's ability to deliver.** Failure to meet promised milestones, business cases that show less and less profitability and constantly slipping launch dates erode trust. The delays make it a lot more difficult for your investors, lead customers and partners to support you when the product is ready for launch.
- **Once your product ships, you have nowhere to go next.** You've just learned a lot, but that knowledge is not extensible into the next product in the product family. The developers know only about the decisions they made that worked or didn't work for this product. They know nothing about how to meet goals for product enhancement, cost reduction or quality improvement because they didn't gather this data, even when it would have been easy and inexpensive to get. The next product will be just as hard to get out the door.

In all of these areas, long, slow learning cycles increase time and cost, while they decrease quality and customer satisfaction. They lengthen the distance between you and your new product. The cumulative effects from all these barriers can kill an idea before it ever reaches a customer's hands, even if it's the best idea you have ever had.

You can choose to develop your product this way and just plan to spend lots of time dealing with the consequences. Or you can break up these long learning cycles to accelerate them.

Break Apart the Learning Cycles to Get Your Product to Market Faster

When route finding, we go a short distance, notice where we are and plan the next short distance. We look for paths of least resistance to our destination, one section at a time. We take just a little time scouting out the terrain ahead of us, to find the mountain passes, the river crossings and any trails that might already have been cleared to get us at least part of the way. We check in frequently to make sure we're on the right track, that we're still scouting ahead for obstacles and always looking for opportunities to take a shorter route to our destination.

Rapid Learning Cycles are a synchronized set of experiments to remove uncertainty before key decisions need to be made in a product development program. One Rapid Learning Cycle is two to eight weeks of focused work to help the team make better decisions when uncertainty is high and so is the impact of the wrong decision. It's a series of scouting trips to plot the shortest distance between you and your new product.

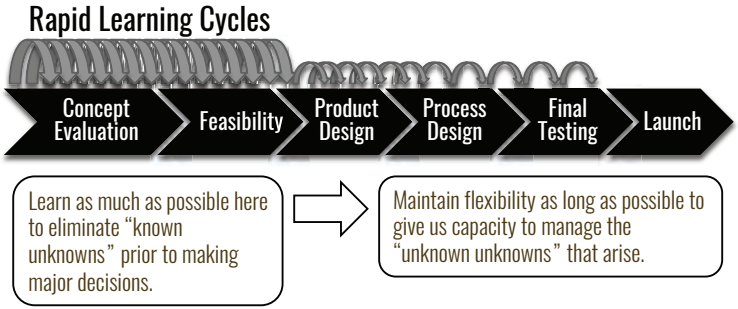

Figure 1.3: Rapid Learning Cycles

The key word is "rapid." I know that startup teams and innovation groups operate with severe shortages of money and time. This is not the time to do in-depth market studies and extensive technical investigations to make sure that every "i" is dotted before going to first release. In fact, in some circumstances, your best Rapid Learning Cycles will be built around limited releases to paying customers, so that you can evaluate their re-

sponses before finalizing decisions. Even on very long projects, such as new drugs or defense systems, you still need short learning cycles—twelve weeks is an absolute maximum.

Rapid Learning Cycles Pull Learning Earlier and Push Decisions Later

Rapid Learning Cycles work because they pull learning earlier and push decisions later. Teams delay making major commitments until they have the knowledge to make those commitments with confidence.

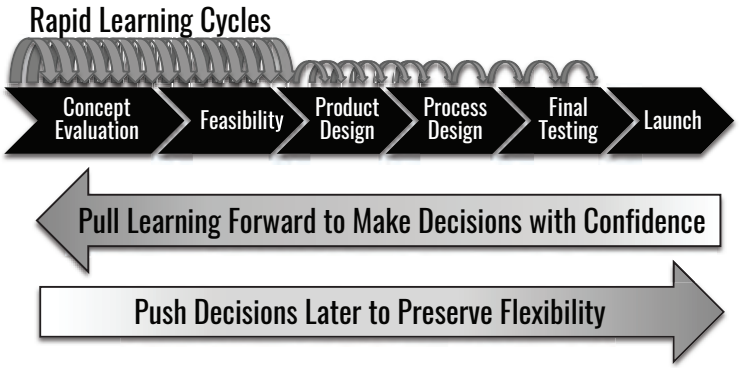

Figure 1.4: Pull Learning Forward—Push Decisions Later

Rather than write up detailed specs in Phase 1 of the PDP, the team lets key requirements float until they understand what they can deliver. Instead of pursuing one idea at a time for a tricky technical problem, the team investigates several possible answers at once. Rapid Learning Cycles provide a framework for determining the right time to make decisions, and the learning needed to make them.

Product development teams make thousands of decisions in the context of a development program. It would take an infinite amount of time

to learn everything perfectly in order to make every decision absolutely correctly the first time. This is not the goal of Rapid Learning Cycles.

The goal of Rapid Learning Cycles is to focus the team's early attention on the major decisions that they must get right if they want to release a great product. Then the framework helps them structure the knowledge-building work they need to do in order to make the best decisions they can. They spend a little more time learning in order to spend less time spinning in Build–Test–Fix to repair all the fallout from decisions made too early.

When I asked the team of managers that I mentioned earlier to draw a Future State of product development, they didn't draw a straight path. But they did show an "advance team" of learners who were scouting for the best places to cross the rivers, navigating around blind canyons and digging tunnels under cliffs so that teams didn't have to go over them. The advance team scouted the terrain and mapped the path of least resistance. The rest could hit the ground running. The managers represented the team's forward motion with scribbled circles—Rapid Learning Cycles.

How Rapid Learning Cycles Get Your Best Ideas to Market Faster

Rapid Learning Cycles get your best ideas to market faster by eliminating many of those long, slow learning cycles. When you have eliminated the worst of them, these are the benefits you'll see.

Set Aggressive Dates—and Hit Them

Many product development groups and most startups deliver products to the market later than they predict—sometimes very late. This wreaks

havoc in the downstream organizations, such as Operations, Supply Chain, Marketing and Sales. They struggle to launch a product that may have already missed its window of opportunity. Even if the product is still competitive, these groups see all the opportunities missed: trade shows where the product couldn't be shown, major customers that went with a competitor's product, a competitor's inferior product becoming established as the standard while the team's product languishes in prerelease mode.

Projects are usually late because the teams planned—or were told to plan—for everything to go as expected, and then the unexpected happened. Unanticipated problems trigger a lot of work that the team did not plan to do. The team may need to change decisions that are fully embedded in the product design and difficult to undo. Under the best circumstances, the amount of work added goes up dramatically as the product approaches completion, and yet this is exactly when these unexpected things tend to happen.

Traditional mitigation strategies for late design changes make things worse. Tight change control adds overhead that makes it difficult to move quickly when the team finds a problem. Locking down requirements early increases the likelihood that they will change. Design "freezes" operate the same way: forcing decisions to freeze early increases the chance that a given decision will be wrong. Forcing development teams to set aggressive dates too early drives them to take shortcuts that will hurt them later.

A product development group should be able to deliver its products on time—if the group does not set the final launch date until midway through the development process. At that point, the team members should know enough about what they understand and what they still need to learn in order to deliver a finished product that is good enough for the market it's going into. They should be able to establish a launch date that they can stick to. The more you can push decisions later, the more predictable your launch dates will be.

Rapid Learning Cycles increase teams' ability to deliver on time by directly attacking the root causes of late-breaking issues: the lack of learning up front before decisions need to be made, and the pressure to make decisions before the team is prepared to make them. By accelerating the learning process, teams get to correct errors before they become major issues, and the decisions they make are more likely to hold up under the pressure of integration in late development. By the time a team has learned enough to make the most important decisions and understand the risks they take for the things they have not yet learned, they are prepared to set an aggressive launch date—and hit it.

Fail Faster to Move On Faster

Some ideas have momentum, even when they're not good ones. Once a startup has its first round of funding or a development team has been assigned, the product takes on a life of its own. Any idea requires some things to be done that will be easy to do and some things that will be hard to do. If the idea is not going to work, the problem probably lies with the things that are hard to do. But since we are human, we tend to do the easy things first and then see all that effort wasted when a hard problem comes to the surface that requires a major redirection.

In an established company, Gate Review meetings are supposed to weed out the products that are likely to fail, before they are approved to go on to the next phase. At these meetings, a team of managers reviews the team's progress and the current state of the business case. In practice, few products get killed at these meetings. Even products with major problems get the waivers they need to continue to the next phase.

Rapid Learning Cycles help you get your best ideas to market faster by eliminating your weak ideas sooner, so that you don't waste time with them. If your idea—or your company—is completely new to the world, early Rapid Learning Cycles will focus on viability tests that can be done

before you commit to anything that would be expensive to redo. This is especially important if your product is based in the real world, as opposed to the virtual world of software and Internet applications. A physical product requires investment in things that are expensive to change if you need to pivot after they are done.

Bring the Vision and Execution Closer Together

Great ideas fail in execution, and there is nothing more disappointing to an innovator than a great idea that was so poorly built that it never stood a chance. We want the final product to exceed all the promises we make to ourselves, our families, our investors and our customers.

Established companies have brand promises they need to protect, and sometimes they incorporate tools such as Failure Modes and Effects Analysis to try to eliminate quality problems within the design. But these efforts usually come too late, when the decisions embedded in the design are difficult to change. Products that don't meet quality standards in an established company tend to get stuck in Build–Test–Fix loops until they reach an acceptable quality level, which can be months or years after the product needed to be on the market to be competitive.

Rapid Learning Cycles bring execution barriers to the surface, where teams can tackle them head on. A team is not forced to commit to a fixed idea about what the product is and what quality level it needs to have until the team is ready to make commitments. If an insurmountable execution barrier leads to the need to pivot, there is a lot less momentum to overcome.

Build a Great Product You Can Afford to Make

A late product is often a compromised product that costs more for production, sales and support until the company discontinues it. It may be difficult to make, with tolerances that have to be too tight for the manufacturing process. It may require more manual assembly because it's too complex for automated manufacturing methods. It could have reliability problems that trigger extra field service calls. It could be a tough sell because the customer is hearing bad things about others' experiences with it, and sales reps may waste a lot of time on the phone with dissatisfied customers that they could have used to find new ones.

When a product is already late, or just a few weeks from launch, teams naturally focus on finding the fastest solution instead of the least expensive one. They may be given permission to make compromises that would have been unacceptable just a few weeks before. Often there isn't time to adequately test the solutions to ensure that they fix the problem before they get incorporated into the product. These late fixes can trigger new issues of their own.

Established companies deal with this problem by adding additional reviews and sign-offs to the process—but always too late to make a difference in the final product without a lot of redesign work. Instead, these mitigations generate a lot of extra work just when the team doesn't have any time to spare. Sometimes groups try to isolate new product development teams from "current product engineering" to lower costs and fix quality problems once the product is launched. This just disconnects the new product developers from the feedback they need to make the next product better, and eliminates any incentive for them to try.

Rapid Learning Cycles prevent these cost issues from occurring in the first place. The team is focused on learning the most effective way to build the product. The structure provides natural points for feedback from production partners, suppliers and support reps so that the team can take

advantage of their knowledge to build a better, lower-cost product from the beginning.

Set Yourself Up for a Fast Version 2.0

Once you have the first product out, it stands to reason that the next product will be a lot easier. But sometimes teams find that they've put everything they know into the latest new product, and they have no idea where to go for Version 2.0. Sometimes Version 1.0 is so compromised that the next version will have be rebuilt from the ground up.

This is especially true if you work in an established company that has traditionally had long product development timelines. If it's going to be three years before the next product comes out, every good idea has to get into this product—or wait another three years.

But every good idea adds complexity to the product and increases the likelihood that something will go wrong somewhere, especially if the team adds them without taking the time to learn about them first.

Startups are often focused on getting their first product out the door, and their teams are exhausted when it finally ships. If they have focused on delivering the best solution, instead of an MVP, it may not be obvious to them how to evolve the product. They may not even have allowed themselves to think that far ahead.

When you are first trying to turn a great idea into a finished product, you naturally want to focus on what you need to do—and only what you need to do—to make your vision a reality.

Everything else seems superfluous, a waste of time and money. Yet that leads the team down a narrow alley without a lot of room to maneuver if something goes wrong. Over time that erodes an organization's confidence in its ability to deliver growth through innovation.

Rapid Learning Cycles encourage test-to-learn vs. test-to-ship. A little bit of experimentation in the early phases of the project can help the team preserve more flexibility later on.

It also has the benefit of giving the team a lot of places to go with the next version of the product. The team may add a technology that was promising but not quite ready, or use a material that was too expensive for an MVP, but would be interesting in a premium version of the product, or build a feature that needs a fresh user interface if it's going to work smoothly.

Team members will not only be excited to get their first product shipped, but eager to come back to work on Monday and get started on the next version. They will grow in their ability to deliver innovative solutions and then build on those to create product families and even new lines of business. They will have greater confidence in their ability to turn ideas into products and that confidence will spill over into the rest of the business.

Rapid Learning Cycles Build Confidence in Product Development's Ability to Innovate

When product developers have a history of missed timelines, quality problems, high costs and disappointing features and/or performance, their business partners lose confidence in them. The partners, and sometimes the product developers themselves, stop believing in the company's ability to innovate. They take a wait-and-see attitude about anything new that emerges from R&D, and they tell key customers to wait to purchase a new product "until they get the bugs worked out."

Long, Slow Learning Cycles Erode Confidence

When product development is one long, slow learning cycle, business partners see disappointing products that require a lot of firefighting in late development. These fires burn all of the downstream functions. Production can't meet its production plans. Supply Chain has to deal with change orders and failure to meet commitments to vendors. Marketing can't launch products effectively, and sales teams don't get the product information they need in time to meet their sales goals. It's no wonder that other functions begin to lose faith in R&D's ability to deliver, and they build plans that assume that R&D will be late.

Your brand equity and your customers' expectations can be seen as barriers to innovation, since the need to serve existing customers tends to kill immature technology rather than incubate it. Your partners may look jealously on the freewheeling environment that allows startups to try one new thing after another until they get it right.

Lack of Confidence Leads to Extreme Measures

We know that our business partners lack confidence in product development when they refuse to stake their own success on our ability to get the product out the door. Instead, there is a lot of finger-pointing between R&D's leaders and other functional leaders. This is sometimes so bad that business teams succumb to the siren song of outsourced product development or "open innovation"—both models built on the assumption that outsiders can deliver better products than the company's own engineers.

Other traditional solutions include holding R&D accountable with performance bonuses for on-time delivery, locking down requirements early and refusing to allow changes, and sometimes firing people who have led failed projects. But performance bonuses (or the lack of them) and firings simply create a climate of fear that impedes innovation as everyone

plays it safe. Locking down requirements early just increases the likelihood of delivering a product that no one wants to buy.

Some experts recommend separating innovation teams from the rest of the company to keep the established organization from squashing compelling yet immature investments—so-called "skunkworks" teams. These measures keep middle managers from canceling the projects too quickly.

Other experts advocate for focusing the engineering teams on improving existing products and looking to the outside for more innovative products: design firms, outsourced engineering staff and strategic acquisitions to purchase new technologies the company can't seem to develop internally. These strategies have their place in some situations. But only a small number of truly breakthrough products have followed this path—not nearly enough to justify the disruptions they cause.

Products produced using this disconnected process are not more likely to get to market, because instead of being acclerated, learning slows down. The innovations suffer from lack of acceptance when they are ready for commercialization. The products don't integrate well into the rest of the company and can't leverage the company's core functions. If they don't get spun off, they die a slow death.

More typically, a compelling innovation collapses under the weight of all the work required to ensure that the product meets company standards once the product idea encounters the organization's internal systems, and never sees the light of day. Fortunately, there's a better way.

Raid Learning Cycles Build the Capacity for Innovation

We've found that Rapid Learning Cycles help avoid a lot of the problems that established companies have when they try to innovate. They help the innovation teams mature a technology until it's ready to release with the company's reputation behind it. They provide structured timeframes that prevent innovators from getting thrown off-course by shifting

priorities and the needs of the current business. They help teams leverage the relevant aspects of the organization's knowledge base without being limited by it.

The emphasis on early learning encourages broad exploration that supports teams as they develop creative solutions to customer problems. Whether you seek growth by finding new customers for your technology or by finding new solutions for your existing customer base, your teams will be able to identify the knowledge they can leverage and the knowledge they need to build.

When your Innovation Process combines structure, agility, the ability to learn fast and the ability to know what to learn, you have an engine to drive the most disruptive innovation over all the hurdles that the status quo puts in your way.

Better Engagement and Results Rebuilds Confidence

The first step to restored confidence is engagement. Rapid Learning Cycles increase the organization's confidence in R&D by engaging partners early and often in cross-functional learning events that make the team's progress visible. The framework provides a structure for getting critical input from these downstream partners at the point in development where it will be the most useful. It builds a sense of shared responsibility for the product's success that defuses all the finger-pointing, as the functional group representatives become part of a team committed to delivering a great product.

The true benefits come when your teams experience success on a regular basis and so do your business partners. When teams uncover obstacles early in a program, they can remove them before they become sources of conflict. Partners get fewer nasty surprises at the end. The products themselves benefit from the ability to discern what knowledge can be reused, what needs to be extended and what needs to be created.

As the teams and R&D management gain more confidence, they become more tolerant of risk and more willing to explore ideas that are outside their comfort zones. They know they have the skills to evaluate ideas to find the good ones, and they know they can build the knowledge they need to reduce the risks of a good idea that may have been outside their comfort zone.

It will be easier for them to explore fields of inquiry that will lead to breakthrough innovations, because they will know that they have what it takes to understand an idea before they commit the organization's resources to the expensive work of commercializing it.

Pull Learning Forward and Push Decisions Later

If you take away just one thing from this book, let it be this: the more you can pull learning forward and push decisions later, the faster your ideas will get to the market.

This is not easy, whether you are working by yourself in a garage or as part of a team inside one of the world's largest companies. All of our natural instincts as innovators and entrepreneurs tell us to get moving.

Our hands burn with the desire to build stuff that will bring our visions to life. These are good instincts. They motivate us to go into the garage or the office every day, to bring our ideas to life and get them into the hands of customers who will appreciate them.

These instincts just need a little redirection. Before diving into your first build, take some time to review the ground between you and your vision. Ask yourself what you already know and what you need to learn to deliver the product you've promised yourself to deliver.

Then build those tunnels and bridges by focusing on what you don't know first, because those are the things that slow you down and make the easy things a lot harder than they need to be.

Instead of building the perfect system, identify the smallest piece you can build in order to learn something you need to know. Instead of delivering the perfect product that provides a full solution, deliver one piece that fulfills a specific customer need.

Use the tools we have for rapid prototyping in the virtual and physical worlds to conduct experiments and test variations, rather than throwing together a product that will have to be rebuilt later.

It's Your Product—
The Responsibility Starts with You

Most people learn how to do product development on the job. While engineering programs teach the technical side of design, and some business schools teach courses on innovation or entrepreneurship, almost everyone learns how to develop products by doing it, sometimes alongside more experienced engineers and sometimes with peers.

This is why established companies go so readily to solutions that lengthen the distance from idea to market. By pulling decisions earlier and increasing management oversight, they get the illusion of control. By exhorting people to go faster, rewarding heroic efforts and excessive overtime, and setting aggressive dates early in a project, they encourage people to take shortcuts on knowledge building that will trigger loopbacks later. Startups are more likely to throw out the rule book, but the effects are the same, and come with a higher personal cost to the innovators.

If Rapid Learning Cycles were easy and intuitive, we would all use them. Instead, they ask you to do some things that are uncomfortable:

- Invest more time and money in the "fuzzy front end" and knowledge-building experiments, to save time and money at the end.
- Delay decisions about things like final specifications, customer requirements and technical solutions in order to ensure that these decisions get made right the first time.
- Pursue multiple alternatives when you'd prefer to explore only the idea that looks the best on the surface.

If you want to get your ideas to market eventually, you can always do things the way people have always done them. No one will get fired in a Fortune 500 company for using the standard PDP, as long as things don't get too far out of hand. And most startups are supposed to fail anyway.

If you can get comfortable with the changes required by the Rapid Learning Cycles framework, however, you are well on your way to building a product development organization that can deliver products to the market faster, with fewer resources, at lower cost and with better quality. You can see your visions come to life and be even better than you imagined them.

When teams pull learning forward and push decisions later, they make better decisions that help them go straight from San Francisco to Los Angeles without a detour to New York.