

Vol. 28, No. 6
June 2015

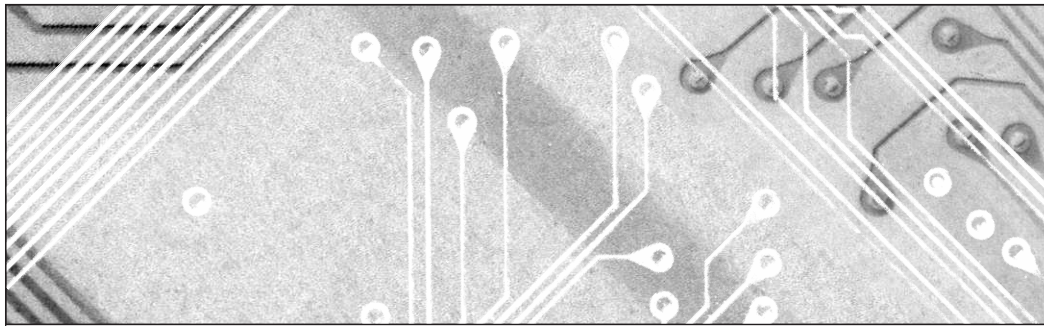
"The more we learn about how software is created and the modern product lifecycle, the more we learn that standardized, rote processes and tools are counterproductive. Change happens quickly, and business needs to respond in kind."

— Jim Benson and
Tonianne DeMaria Barry,
Guest Editors

Practical Management: Lean and Beyond

Opening Statement by Jim Benson and Tonianne DeMaria Barry	3
Managing for Continuous Improvement Using the Improvement Kata and Coaching Kata by Adam Light	6
Managing Complexity: Creating Leaders at All Levels by Esther Derby	14
How Lean Management Systems Can Enable Agile at Scale by Steve Bell and Karen Whitley Bell	19
Goal #1: From Activity to Action by Arne Rook	26
Practical Lean Software Development for Microenterprises by Andrea Janes	30

NOT FOR DISTRIBUTION
For authorized use, contact
Cutter Consortium:
+1 781 648 8700
service@cutter.com



Cutter IT Journal

About Cutter IT Journal

Part of Cutter Consortium's mission is to foster debate and dialogue on the business technology issues challenging enterprises today, helping organizations leverage IT for competitive advantage and business success. Cutter's philosophy is that most of the issues that managers face are complex enough to merit examination that goes beyond simple pronouncements. Founded in 1987 as *American Programmer* by Ed Yourdon, *Cutter IT Journal* is one of Cutter's key venues for debate.

The monthly *Cutter IT Journal* and its companion *Cutter IT Advisor* offer a variety of perspectives on the issues you're dealing with today. Armed with opinion, data, and advice, you'll be able to make the best decisions, employ the best practices, and choose the right strategies for your organization.

Unlike academic journals, *Cutter IT Journal* doesn't water down or delay its coverage of timely issues with lengthy peer reviews. Each month, our expert Guest Editor delivers articles by internationally known IT practitioners that include case studies, research findings, and experience-based opinion on the IT topics enterprises face today — not issues you were dealing with six months ago, or those that are so esoteric you might not ever need to learn from others' experiences. No other journal brings together so many cutting-edge thinkers or lets them speak so bluntly.

Cutter IT Journal subscribers consider the *Journal* a "consultancy in print" and liken each month's issue to the impassioned debates they participate in at the end of a day at a conference.

Every facet of IT — application integration, security, portfolio management, and testing, to name a few — plays a role in the success or failure of your organization's IT efforts. Only *Cutter IT Journal* and *Cutter IT Advisor* deliver a comprehensive treatment of these critical issues and help you make informed decisions about the strategies that can improve IT's performance.

Cutter IT Journal is unique in that it is written by IT professionals — people like you who face the same challenges and are under the same pressures to get the job done. *Cutter IT Journal* brings you frank, honest accounts of what works, what doesn't, and why.

Put your IT concerns in a business context. Discover the best ways to pitch new ideas to executive management. Ensure the success of your IT organization in an economy that encourages outsourcing and intense international competition. Avoid the common pitfalls and work smarter while under tighter constraints. You'll learn how to do all this and more when you subscribe to *Cutter IT Journal*.

Cutter IT Journal®

Cutter Business Technology Council:
Rob Austin, Ron Blitstein, Tom DeMarco,
Lynne Ellyn, Israel Gat, Vince Kellen,
Tim Lister, Lou Mazzucchelli,
Ken Orr, and Robert D. Scott

Editor Emeritus: Ed Yourdon
Publisher: Karen Fine Coburn
Group Publisher: Chris Generali
Managing Editor: Karen Pasley
Production Editor: Linda M. Dias
Client Services: service@cutter.com

Cutter IT Journal® is published 12 times a year by Cutter Information LLC, 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA (Tel: +1 781 648 8700; Fax: +1 781 648 8707; Email: citjeditorial@cutter.com; Website: www.cutter.com; Twitter: @cuttertweets; Facebook: Cutter Consortium). Print ISSN: 1522-7383; online/electronic ISSN: 1554-5946.

©2015 by Cutter Information LLC. All rights reserved. *Cutter IT Journal®* is a trademark of Cutter Information LLC. No material in this publication may be reproduced, eaten, or distributed without written permission from the publisher. Unauthorized reproduction in any form, including photocopying, downloading electronic copies, posting on the Internet, image scanning, and faxing is against the law. Reprints make an excellent training tool. For information about reprints and/or back issues of Cutter Consortium publications, call +1 781 648 8700 or email service@cutter.com.

Subscription rates are US \$485 a year in North America, US \$585 elsewhere, payable to Cutter Information LLC. Reprints, bulk purchases, past issues, and multiple subscription and site license rates are available on request.

☐ Start my print subscription to *Cutter IT Journal* (\$485/year; US \$585 outside North America)

Name	Title	
Company	Address	
City	State/Province	ZIP/Postal Code
Email (Be sure to include for weekly <i>Cutter IT Advisor</i>)		

Fax to +1 781 648 8707, call +1 781 648 8700, or send email to service@cutter.com. Mail to Cutter Consortium, 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA.

SUBSCRIBE TODAY

Request Online License Subscription Rates

For subscription rates for online licenses, contact us at sales@cutter.com or +1 781 648 8700.



Opening Statement

by Jim Benson and Tonianne DeMaria Barry

As we have worked with or spoken to modern business people, we have noticed an overwhelming frustration exists within the world of software. *Why can't we accurately estimate? Why is planning so difficult? Why does effective communication present such a challenge? Why is it so hard to collaborate? Why do I have so many meetings?*

As frustration mounts, so too does the desire to end that frustration. This is dividing people into two mutually counterproductive and polarized groups:

- Group One wants process to plug in and save everyone from everything.
- Group Two responds to this by hating anything having to do with process.

So Group One retains the services of an endless stream of consultants and trainers, while Group Two becomes increasingly annoyed by the endless stream of consultants and trainers. The consulting hordes' suggestions are then routinely lauded by half the room ... and predictably condemned by the other.

Not surprisingly, legitimately good process ideas then fall victim to these tribal struggles. Ideologies develop based on the desire to implement, rather than as a result of actual implementation. Tools are easy to argue about, as they are things and not ways of thinking. In political battles, tools then become the focal point and are hyper-scrutinized. Groups One and Two begin to debate whether or not the tools will produce value that often the company has never even contemplated providing. Then they summarily reject them when they don't supply the phantom value.

As major process families like Agile and Lean roll out to more companies and contexts, they are molded to include ideas and practices never intended to be part of the system. For example, Agile implementations end up adopting tight deadlines and prescribed work, or Lean solutions result in restrictive metrics and unwavering definitions of standard work. Their core messages become strained, diluted, and unrecognizable.

Angry people in meetings will say, "What *is* Agile, really?" or "Is it a better approach for us?" or finally, "Look, I don't care anymore. Just tell me what to do!"

At their core, Agile and Lean are predicated on paying attention and continuously improving both our processes and our products. Tools like kanban, Personal Kanban, A3s, validation canvases, and the like, are spreading Lean thinking. Tools like user stories, timeboxing, and velocity are spreading Agile. This would be all well and good, but as either process family gains acceptance, the focus of adopters increasingly falls on the tools and not the principles or the people actually doing the work.

People are told *what* to do, as they have requested. They are given tools and specific prescriptive paths in which to employ them. They engage in explicit rituals at precise times. But they are often oblivious to *why* they are doing these things. They experience some minor improvements and feel better, becoming convinced that what they are doing is working. They see the tools as the practical side, the theory as the frivolous side. Practicality in management, however, requires that we appreciate both, becoming adept at building systems that include both the implementative and social elements.

HOW DID WE GET HERE?

Historically, software and IT are two areas that cause most project managers and business owners to throw up their hands — they simply cannot believe that software projects can be predictably managed. Often this is blamed on the people doing the work: "Software people are weird," "They're terrible communicators," "They can't answer our questions," or "They just can't estimate to save their souls."

But software developers and IT people in general are just that: *people*. People are not the problem; they are subject to systems that will not abide clear estimates and always include an element of ambiguity. It's like blaming someone for being wet when they get caught in the rain.

The process of making software itself is relatively tameable, but by no means as standardizable as physical work. The more we learn about how software is created and the modern product lifecycle, the more we learn that standardized, rote processes and tools are counter-productive. Change happens quickly, and business needs to respond in kind. Ideally we want to increase predictability, but the best we can hope for is to simply understand what is predictable, minimize some of the unpredictable, and build systems to suit.

W. Edwards Deming notably said, “A bad system will beat a good person every time.” In knowledge work, most of the value creation happens in people’s brains. Their frame of mind is therefore directly related to how well they can complete their job. The systems we create must respect people, build coherent lines of communication, and distribute decision-making authority throughout the organization.

Change and variation are endemic to knowledge work and software development in particular. We are just now learning how to recognize, analyze, and respond to these destabilizing factors. This means that, as difficult as it may be to hear, the software industry is still in its infancy. We will mature when we collectively understand and routinely operate in ways that fully embrace and explore change and variation.

UPCOMING TOPICS IN CUTTER IT JOURNAL

JULY

Robert N. Charette

Mitigating the Risks of Technology Backlash

AUGUST

Lou Mazzucchelli

What’s Over the Technology Horizon?

SEPTEMBER

Curt Hall

How Wearables Will Impact Corporate Technology, Business, Social, and Legal Landscapes

OCTOBER

Vince Kellen

IaaS: Technology, Legal, and Security Implications

LET’S TALK REAL PRACTICALITY

Since we are dealing with evolving products in evolving markets, the practical approach would be to build systems that aggressively learn and improve. This is Agile. This is Lean. But it is also something more. We are not merely interested in manufacturing software or any other product. We are interested in building the right thing in the right way at the right time. We want our product or service to get to market when it will make the most money and delight the most customers.

This requires bold organizations willing to create systems that let software be developed successfully. This requires respecting the professionalism of *everyone* in the building — not just the developers, but UX, design, QA, systems architects, even managers. The creation of good software is not an individual sport, it is a team effort. It is focused collaboration.

This requires all members of an organization to come together and deal with the complex challenges that we face in the creation of value in an uncertain and changing world. There are some key components to a practical management system:

- **Respect for people.** While this seems frivolous or perhaps a platitude, we must practically respect the people that are in the teams producing value for us. This doesn’t mean saying “good job” or giving weekly pats on the back. It doesn’t even mean raises. It does mean building systems that simply let professionals do their jobs in the best ways they can. They need to be able to exercise judgment, identify opportunities and threats, and act when they feel action is needed.
- **Clarity.** Knowing what work is underway, who is doing it, what drives deadlines, what “right” looks like, how to communicate with others, who the customer is, and what shifts the market is experiencing is no longer the job of product managers. This is now vital information of which all members of all teams must have at least a passing understanding.
- **Flow of information.** People can only act on information they have. Information about what work is in flight, who is working on what, what is blocked, what is more complex than estimated, and what the potential risks are is easily communicated to team members, but it is often summarily withheld. Therefore, team members do the wrong things because they simply don’t know better.
- **Management for real.** The replacement of guesswork estimates with actual statistical forecasts removes much of the previous burden and pain from both project management and implementation.

- **Collaboration.** Most knowledge work involves regular occurrences of tasks that are complex. Complex tasks require diverse thinking. They need to be swarmed on or team-solved. Removing the stigma of failure from a person who can't solve a complex task (because the complex task is too complex for any one individual) and replacing it with an opportunity to collaborate and learn results in better solutions and healthier learning cultures.

When we set out to implement Agile, or Lean, or any of the other flavors of management, we must first and constantly ask, "Am I building a practical system that will let professionals produce the most value for the customer and the company and not cause unexpected internal strife? In their own ways, the authors of the five articles in this issue of *Cutter IT Journal* ask this question and show either theoretically or directly how real systems engage real people.

Adam Light begins by providing a way to achieve the Agile practitioner's goal of *being* Agile — and darned if it doesn't turn out to be *doing* Agile through the use of the "improvement kata" and the "coaching kata." By creating systems that support learning and improving one's ability to improve, the focus of both an organization and its teams becomes more collaborative and more streamlined through increased positive social interaction. Learning is not only retained, it is utilized. Learning is not only expected, it is anticipated. Light tells us how to scale learning through distribution.

Next, Esther Derby discusses the natural tendency for organizations to restrict information flow as they grow. The tendency to consciously or unconsciously hoard information for personal gain or comfort inhibits value creation by keeping some areas of a company information-deprived. She provides a framework to not only help companies make the most of the leadership they have, but to foster it throughout all corporate levels by providing the appropriate clarity, conditions, and constraints.

In our third article, Steve Bell and Karen Whitley Bell show how Agile is attempting to scale from a small, inwardly focused, team-based set of tools to an organizationally inclusive set of tools. They and other Agile thinkers are finding that Lean is a necessary adjunct to Agile ideas in order to operate at scale, particularly the Lean concept of the value stream. The authors offer a value stream perspective on various software development challenges and introduce two Lean management system practices "that promote the ability of the enterprise to foster a value stream perspective and improve overall value and performance."

Next up, Arne Rook describes how his company, Jimdo, actively applies systems thinking specifically to foster alignment and shared purpose — even in a team-based company with very little central planning. Jimdo had discovered that as it grew, its teams were losing their sense of "being on the same page," and activity was becoming a substitute for action. In response, they developed Goal #1, which leverages the Lean concept of work in progress and systems thinking's notion of global optimization to tackle goals characterized by a high degree of uncertainty and critical strategic importance.

Andrea Janes closes the issue with a data-based approach to Lean software development that enables microenterprises to eliminate waste and produce more value. Applying the ideas of Mary and Tom Poppendieck, Janes shows how the automatic collection of data on development activities can help software teams in small companies reap the benefits of Lean software development — reduced waste, higher software quality, improved processes, faster delivery — for a relatively modest investment.

The articles in this issue represent years of experience by the authors in creating management systems that achieve their efficiencies by allowing knowledge workers to do what they do best: solve problems. We hope they serve as an inspiration to you in your journey toward better, more effective management.

Jim Benson, CEO of Modus Cooperandi, specializes in Lean project management and the management of knowledge work. He is the creator of Personal Kanban and, with Tonianne DeMaria Barry, coauthor of the book Personal Kanban, which won a Shingo Research Award for Excellence in 2012. He is the 2012 winner of the Brickell Key Award for excellence in Lean thinking. For the past two decades, Mr. Benson has worked at uncovering ways for groups to find clarity in unpredictable and amorphous knowledge work environments. Since starting Modus, he has helped the World Bank, NBC Universal, the United Nations, Spotify, Riot Games, Comcast, R.W. Baird, and others improve their kanban systems, implement collaborative solutions, identify and implement improvements, and create more innovative cultures. He can be reached at jim@moduscooperandi.com.

Tonianne DeMaria Barry is a partner at Modus Cooperandi, a Shingo-award winning author, systems thinker, photographer, and historian. She looks beyond the apparently obvious problems and sees their real social and systemic root causes. Ms. Barry is the coauthor (with Jim Benson) of Personal Kanban and the upcoming Why Kanban Works. She is cofounder of Kaizen Camp, the continuous improvement un-conference with events held worldwide. Passionate about the roles that collaboration, value creation, and happiness play in "the future of work," and appreciative of the ways in which Lean thinking can facilitate these ends, Ms. Barry works with clients globally to create cultures of continuous improvement via business environments that are more humanistic than mechanistic. She can be reached at tonianne@moduscooperandi.com.



Managing for Continuous Improvement Using the Improvement Kata and Coaching Kata

by Adam Light

INTRODUCTION

Agile teams that sustain high productivity tend to manage their work visually using simple but effective tools. Through disciplined use of technical practices such as test-driven development, continuous delivery, and emergent design, they deliver high-quality software frequently and at a sustainable pace. Because these teams work closely with knowledgeable customers, they receive rapid feedback on the value of their work, adjusting their focus naturally to supply more of what customers want. Trust builds, mutual respect grows, and team morale remains high.

But whereas a single high-performing team can often succeed through the efforts of a few individuals, the organizational challenge increases greatly as complexity increases through the addition of people, functions, and activities. Even in organizations where multiple teams have been successful, sustained high performance over time is exceedingly rare. Sooner or later, changes in people, technology, or the marketplace disrupt even the most finely tuned organization. When disruption occurs, reestablishing the conditions necessary for team success requires hard work — and disruption occurs

continuously! As the Red Queen famously told Alice in Wonderland: “It takes all the running you can do, to keep in the same place.”

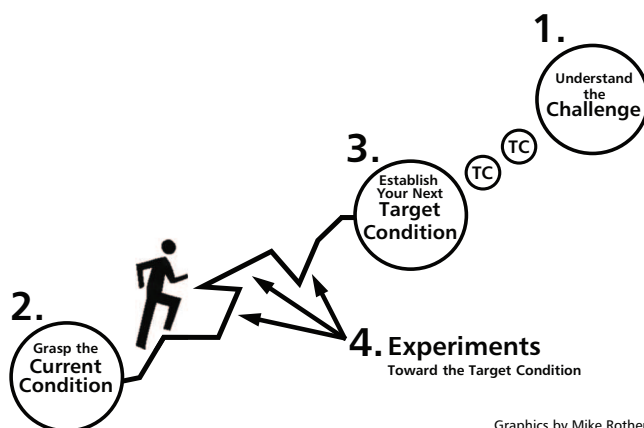
As a method for practicing continuous improvement on a day-to-day basis, the *improvement kata* complements Agile methods and integrates with familiar Lean tools. While applying the scientific method to process improvement is nothing new to those familiar with Six Sigma and similar process improvement methods, breaking improvements down into small parts and implementing those parts using very short planning cycles proves effective for all the same reasons that Agile methods work well in product development and delivery.

A kata is a practice pattern used to develop a set of skills. Leaders at any and every level of an organization can use the improvement kata to practice continuous improvement.¹ Following four simple steps (see Figure 1), they:

1. Understand the direction or challenge they are striving for
2. Grasp the current condition using facts and data about current outcomes and the current process
3. Establish a target condition that describes the next step on the way to the challenge
4. Iterate rapidly toward the target condition using small, frequent experiments

By following the pattern repeatedly, practitioners gradually develop a new mindset as they begin to apply the scientific method to build a deeper understanding of their own work processes, gathering and using data to adapt theories and guide improvements.

Where Agile methods have been successful, people tend to find the improvement kata pattern intuitive and easy to learn, recognizing it as “Agile for process improvement.” Those who already measure their processes typically find immediate opportunities for improvement from their existing data. When individuals at multiple



Graphics by Mike Rother

Figure 1 — The four steps of the improvement kata. Setting fixed dates for achieving each successive target condition helps emphasize small, frequent improvements.

levels in the management hierarchy begin to practice the *coaching kata* consistently, a collective understanding of the work system gradually emerges. The coaching kata is a practice regimen designed to build and sustain mentor-mentee relationships around continuous process improvement. Managers follow the coaching kata to teach the improvement kata pattern as part of daily work, so that it becomes part of an organization's culture. In contrast to improvement initiatives characterized by top-down implementation of "best practices," people who experience the coaching kata and improvement kata increasingly recognize where one size does not fit all.

TOWARD A NEW WAY OF WORKING AT ABC CORP.

A good way to understand the workings of the improvement kata and coaching kata is to follow their first use in a typical IT organization.² The following narrative describes experiences inside "ABC Corp.," a composite organization I'll employ to illustrate how using the practices produces organizational learning that leads toward a new way of working.

ABC Corp. is an established firm that serves both consumers and businesses. Like almost every modern enterprise, software and information technology form an increasingly critical element of ABC's brick-and-mortar business. In addition to the back-office and front-office systems that help ABC organize its employees and serve customers, ABC builds and maintains a portfolio of Web and mobile applications that enable customers to serve themselves by interacting directly with ABC's core systems. ABC began using Agile methods about five years ago and now supports about 25 Agile teams.

The Case for Change at ABC

Ben manages a group of ScrumMasters at ABC who support teams that develop and maintain customer-facing products and systems. As owner of the Agile software development lifecycle (SDLC), Ben is responsible for process improvement. And his internal business clients tell him they see a lot of opportunity for things to get better! ScrumMasters and Agile teams track their progress using a range of metrics, but clients complain that the most important releases don't arrive predictably. Lack of IT productivity constrains client business goals, and in the few years since Agile methods became the norm, clients have stopped perceiving improvement.

Ben recognizes that while individuals and teams are working hard to improve, the organization as a whole

has plateaued. After learning about the improvement kata at a conference, he becomes excited by the potential to optimize globally by linking local improvements together.

The Advance Group

Ben attends an improvement kata training class. He then showcases the improvement kata at ABC by scheduling informational webinars for interested parties. He finds strong interest and support from his direct supervisor and several peer managers and receives approval to begin a journey of improvement within his own department.

Ben joins with two ScrumMasters, his boss Tim (the director of applications), and a peer manager in the customer applications group to form ABC's improvement kata "advance group" (see Figure 2). The advance group is chartered to initiate use of the improvement kata and coaching kata and then to sustain change in the organization toward a new way of managing people. The advance group hires an outside consultant to help them prepare themselves and plan their work.

The advance group begins to learn by attending a three-day kata training course that includes an immersive experiential introduction followed by a visit to a company that already practices the improvement kata and coaching kata. Members of the group observe and participate in daily improvement kata coaching meetings and ask lots of questions; they are amazed to see teams making and validating process changes within the span of a single coaching cycle.

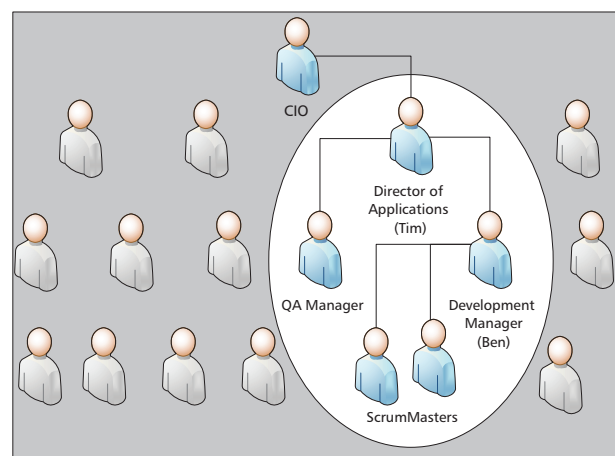


Figure 2 — Makeup of the ABC advance group.

After building their own coaching proficiency, the advance group can help others learn while also working actively to sustain organizational change toward a new way of managing people.

Setting Initial Target Conditions

Returning home, the ABC advance group holds a workshop to establish their first target conditions. They focus on the flow of maintenance releases to a key customer support system. In the existing release process, a dedicated development team feeds work to a separate team that conducts acceptance testing on maintenance releases, which occur every six to eight weeks. Clients and developers alike are aware that the individual changes and fixes involved often require only a few hours to develop, and they feel strongly that releases should get finished faster. The proposed improvement lies in familiar territory. Members of the advance group count themselves as experts in the selected area, and the group agrees unanimously that employees and customers will perceive reaching this target condition as a “win.”

The advance group divides into two groups with a shared focus on different parts of the same process. One subgroup focuses on the development team’s activities, while the other focuses on the testing and release activities. Tim commits to participate as a member of both subgroups. Each subgroup then meets to prepare an improvement kata “learner’s storyboard,” which is similar in form and purpose to an A3 problem-solving report (see Figure 3).³ The storyboard is a big, visible

chart that displays the overall goal or challenge, the current condition and target condition, a list of obstacles that prevent progress, and a written record of recent improvement steps. Each subgroup develops a specific target condition.

First Coaching Practice

Each subgroup then schedules daily coaching meetings during the portion of the morning when team stand-up meetings traditionally occur at ABC, and each displays their learner’s storyboard in the shared work area of the affected work group. Subgroup members align themselves to the three coaching kata roles (see Figure 4), with the understanding that roles will rotate periodically while they are learning the coaching kata. The first meeting of each subgroup becomes a brainstorming session to list obstacles that prevent reaching the desired target condition.

The subgroup working on development looks initially for a way to reduce release-scheduling delays by an average of two days. At first, they have difficulty seeing how to go about this. Eventually somebody suggests conducting a “5 Whys”⁴ analysis. This leads to an impromptu brainstorming session with the entire development team, and the group identifies several obstacles that delay releases. Challenged by their coach to focus


Focus Process: Developing maintenance releases to customer information system		Challenge: Shorten the average time to release a feature (cycle time) to 5 weeks
Target Condition: Achieve by: June 10 (in 3 weeks)	Current Condition: Outcome Metrics: Releases to production occur, on average, every 6-8 weeks, of which an average of 8 days is caused by scheduling delays.	PDCA Cycles Record 
Outcome Metrics: Reduce delays due to release scheduling issues by an average of 2 days.	Process Metrics: Each release contains an average of 7 major features.	Obstacles Parking Lot <ol style="list-style-type: none"> 1. Some releases require manual work, and so require staffing after hours. 2. Rework due to missing data in release plan. 3. Some releases that aren't treated as "standard" changes could be (?)

Figure 3 — A learner’s storyboard created by members of the ABC advance group. The learner’s storyboard helps make the learning process visible and serves as a focal point for coaching kata activities. The ABC development group brainstorms obstacles and selects “manual work requiring after-hours staffing” as the first obstacle to address.

Table 1 — Initial Plan-Do-Check-Act (PDCA) Cycles Record for the ABC Development Team

PDCA CYCLE RECORD					
Process: Developing Maintenance Releases to Customer Information System			Challenge: Shorten the average time to release a feature (cycle time) to 5 weeks		
Learner(s): Amy (ScrumMaster)		Coach: Ben (Manager)		Current Target Condition: Reduce delays due to release scheduling issues by an average of 2 days. Achieve-By Date: June 10	
Start/End	Cycle #	What Will You Do? (Step)	What Do You Expect?	What Actually Happened	What We Learned
May 22/ May 26	1	Obtain technical information to build automation	Prior release documents and existing automation scripts will contain the necessary information.	Most of the information we needed was available, but we had to consult the IVR administrator.	The IVR release documents are not sufficient for use without system administrators.
May 26/ May 28	2	Obtain missing technical information from the IVR administrators	They will be able to provide information over the phone or in person.	One knowledgeable person was on vacation. His back-up had to walk through the procedures, and it took several hours.	Even the system administrators haven't documented all the steps.
May 28/ June 2	3	Build a proof-of-concept automation in development environment	This should be achievable in one day.	We had to troubleshoot and couldn't finish the work for several days. (We needed help from another group.)	The development environment was not configured identically to the production environment.
June 2/ June 3	4	Run the proof-of-concept for a recently installed change to confirm that it works	The automation will run unattended in the development environment.	After several tries, the automation ran as expected and was confirmed by our (on-team) tester.	Our concept is technically feasible.
June 3/ June 4	5	Run the proof-of-concept on a manual basis for an actual install in production	We will use the script in production, having the administrators invoke it manually.	After obtaining special approval for a weekday release, the script ran successfully.	(1) Our script works in production. (2) A special approval is needed for weekday releases; we will need to change this if we want to do continuous integration.
June 4/ June 8	6	Run the automation with system administrators standing by	We will schedule the automation for the time of the weekend downtime and watch it run.	Automation worked in production as it did in the development environment.	Development environment now seems similar to production!
June 8/ June 9	7	Run the automation without system administrators	We will gain approval to run the automation unattended.	Approval was forthcoming, but we had nothing ready to release!	

Following their experience with the initial target condition, the advance group and the trainer decide to prioritize an overview of Kanban (which includes an introduction to batches and queues) and a module on basic value stream mapping. Advance group members then participate in the training, along with members of both teams whose work processes are being addressed.

As an exercise before the first class session, the subgroups each produce high-level process maps of their own work process. In class, the two groups compare and combine their process maps.

Second Cycle of Practice and Learning

Both groups revise their learner's storyboard. Since neither group fully achieved their first target condition, they both decide to renew their focus on the initial goal. However, in each case there is now significantly more detail about the current condition, and the obstacles have become much more fine-grained.

After three more weeks of (almost) daily practice, the development group has set the stage to reduce release delays. And the testing group has reduced *projected* downtime, but they will both need more data to demonstrate conclusively that they've actually

achieved their goals. The two groups think their improvements will help address the overall challenge.

However, as Tim attends regular coaching meetings for both subgroups, he begins to sense a certain tension between them regarding the improvements necessary to reach the overall challenge. The testing group sees little room for improvement in their own processes, while the development group claims that important stories sit in integration testing for weeks.

The second Lean training class builds on the first. Participants learn more about Kanban and examine the relationship between velocity and cycle time. Tim suggests that the class walk a user story through their combined block diagram from end to end, which produces an important insight. When viewed at a high level, the two groups are following one continuous process, but at a more detailed level, the work is not tracked continuously. As soon as the development group completes a story in their current sprint, they place the story into a "ready for integration testing" stage using their team's tracking system. But when the testing group begins their sprint, they build a release package by examining all the available input material, effectively consolidating things into a larger batch, which they begin tracking

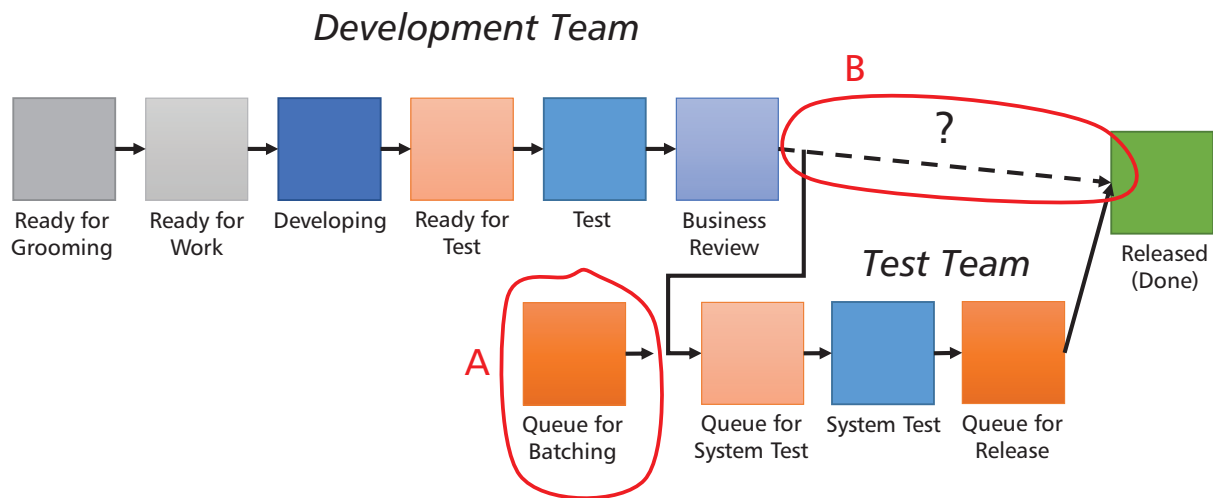


Figure 5 — Block diagram showing the ABC process map after Lean training. Learning to use value stream mapping helps participants in two adjacent processes construct a single process map. Subsequent coaching kata practice leads them to realize that a “hidden” queue lies in the handoff between them (A) and to recognize a shorter potential path for many work items (B).

from that point on. They do not explicitly identify the individual features unless they need to open a defect for one of them.

Tim helps members of the two groups agree to add a new stage to their combined process map that represents a *previously invisible* queue between the two groups (see Figure 5). Exerting his positional authority, he directs that the next target condition focus on measuring discrete work items across the two groups.

A “Chain of Coaching”

Tim now assumes the coach role in both groups. Making the necessary changes to tracking systems takes another three weeks, but both groups reach the target condition successfully, with the result that work items can be tracked from end to end across a single visual planning board.

At this point, only a few work items have been tracked through the full development cycle, but the pattern is already becoming obvious to Tim. He sees that the average story sits for many days in the previously unseen queue. He shares his insight with the course instructor, who adjusts the content of the next class to emphasize the measurement of throughput and to explain how to produce and read a cumulative flow diagram (CFD).

From this point on, improvement kata work begins to gain momentum. Both groups refer to the visual planning board together, and the two ScrumMasters collaborate to maintain a combined CFD. Tim remains in the coach role for both groups, guiding them through a series of target conditions focused on reducing the time

that features spend waiting in this newly identified queue (see Figure 6).

A lack of technical capabilities prevents the groups from moving to a continuous deployment model. However, they are able to make a lot of progress by identifying different types of features and discovering that some of them can skip the integration testing phase altogether. By creating a fast-track procedure for these items, ABC meets and exceeds the overall challenge over a period of about three months!

LESSONS LEARNED

The preceding story shows a pattern of improvement that involves making work visible and addressing local optimization. Many experienced managers and ScrumMasters have seen and worked to address similar challenges. By increasing the pace and frequency of such improvements, the improvement kata and coaching kata can accelerate organizational learning in several ways.

The first accelerant is implementing a pattern for regular practice. The sooner practice begins, the sooner skills and experience will develop. Aligning cadenced target conditions with cadenced sprint planning prevents conflicts. Learning to coach process improvement skillfully involves much more than learning and implementing the patterns, but regular use of the coaching kata gets — and keeps — things moving. Pair programming may be a good analogy from software development. An expert coach can help you make progress, but you can also bootstrap improvement by rotating through the coaching roles. The most important thing is to keep questioning and reflecting.

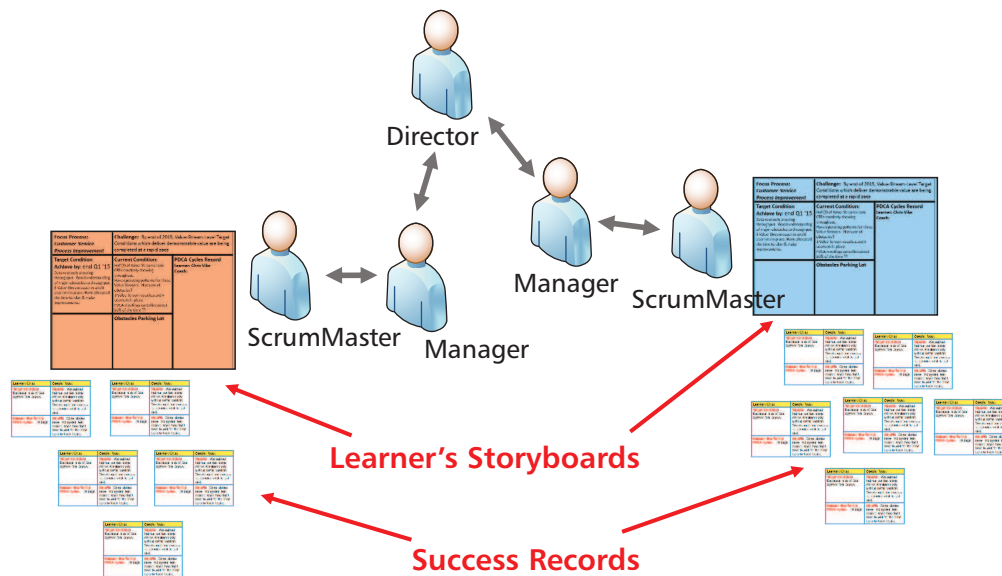


Figure 6 — Chain of coaching, storyboards, and success records. With coaches and learners operating effectively at the process level, Tim focuses on maintaining a department-level challenge that unites work at the process level. He coaches his managers to celebrate successful learning as well as improved outcomes. Making a visible record of each target condition achieved shows the cumulative effect of small, incremental process improvements.

Involving management from the beginning is the second accelerant. In the story of ABC Corp., the two process improvement groups receive guidance and coordination through Tim, their director, who acts as a process coach. Having a wider span of control, Tim can see what has been called the “white space”⁵ between the two groups. Because he practices the improvement kata with two groups every day, Tim learns rapidly and becomes able to formally assume a coaching role in both groups. Senior managers need to develop the best skills so they can eventually coach their direct reports in a “chain of coaching.”

Thirdly, the improvement kata and coaching kata can accelerate improvement by continually fostering a systems perspective. Development processes are seldom defined at a detailed level. Initial efforts, and even whole target conditions, often involve gaining deeper and better understanding of the system. The breakthrough in the story comes when Tim helps learners realize that work is sitting in an “invisible” queue for a long time. Through small conceptual shifts like this, an understanding of the system develops continuously even as the system improves.

In addition to instituting an explicit pattern for learning, aligning coaching skill and practice within the existing management hierarchy, and continually building understanding of the work system, the improvement kata and coaching kata help practitioners to “pull in” new knowledge when — and only when — it can help.

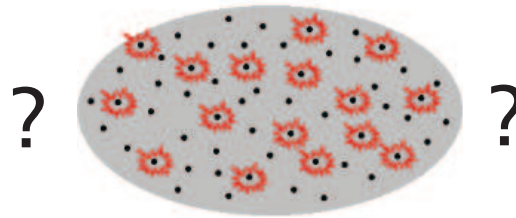
Many process improvement activities bog down in the attempt to learn and apply new analysis techniques or to adopt new work methods all at once — before seeing if they can really help. Beginning with improvements to familiar processes helps get the practice cycle started. Focusing on small, incremental changes helps avoid distractions, as does the recommended practice of reflecting on past improvement cycles to determine what new knowledge may help in the near future. Beginning by creating the ability to practice process improvement consistently and progressing from there to a deeper understanding of Lean concepts usually makes more sense than the other way around.

CONCLUSION

While pundits may praise the virtues of “failing fast,” the fact of the matter is that outside of laboratory walls, our culture and our institutions tend to celebrate success and perfection more often than repeated failure. The improvement kata and coaching kata help address this gap.

The improvement kata complements well-known Agile methods by scaffolding continuous improvement behavior. When asked about continuous improvement, many organizations point to retrospectives and lessons-learned procedures. Without a clear understanding of the current condition and the target condition, however, these valuable activities tend to focus on obstacles

Don't do this...



Do this instead!

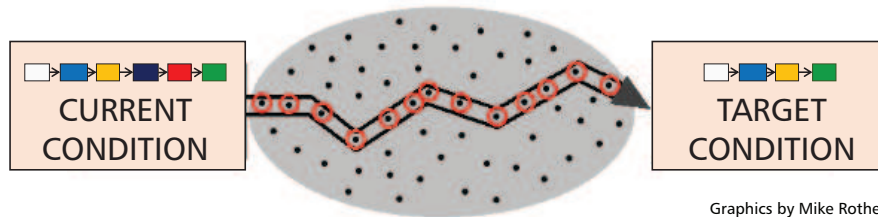


Figure 7 — Continuous improvement with and without current and target conditions.
Process improvement using the improvement kata embodies an explicit goal state.

rather than process changes that lead to specific outcomes over time (see Figure 7).

A frequent meme among Agile authors contrasts *being* Agile with *doing* Agile, where “being Agile” usually means exemplifying ideal values⁶ or principles⁷ and “doing Agile” implies going through the motions without truly changing one’s mindset. Yet this begs the question of how a new mindset grows among members of an interdependent group — such as multiple development teams or the development managers in a given organization — who must all shift the way they work in order for any to exemplify the new way of thinking. Where accomplishment requires command of multiple interrelated skills, obtaining fluency in a new mode requires sustained practice in the work environment. In fact, *being* Agile actually requires practice *doing* Agile, and the improvement kata enables the practice that is needed to learn.

ENDNOTES

¹For a thorough description of the improvement kata, see: Rother, Mike. *Toyota Kata: Managing People for Improvement, Adaptiveness, and Superior Results*. McGraw-Hill, 2010.

²The authoritative general source on the improvement kata and coaching kata is Mike Rother’s *Improvement Kata Handbook*, which can be found at the Toyota Kata (TK) website (www-personal.umich.edu/~mrother).

³An A3 report is a simple storyboard that tells the story of a problem-solving or improvement event on a single sheet of paper with the problem on the left and the solution on the right. (The international A3 paper size of 297x420 millimeters is closest to the 11x17-inch paper used in the US.)

⁴The 5 Whys technique explores the cause-and-effect relationships underlying a problem by repeating the question “Why?” until the root cause is identified.

⁵Rummler, Geary A., and Allan P. Brache. *Improving Performance: How to Manage the White Space on the Organization Chart*. 3rd edition. Jossey-Bass, 2013.

⁶“Core Scrum.” Scrum Alliance, 15 August 2014.

⁷“Lean-Agile Leaders Abstract.” Scaled Agile Framework, updated 11 May 2105.

Adam Light is Management Consultant, Founder, and Principal at SoTech Advisors, where he helps technology leaders apply Lean and Agile methods to increase productivity, deliver customer value sooner, and improve the lives of their employees. Mr. Light has more than 20 years of technology experience. He began his career as an application developer before becoming a manager of projects and people. Mr. Light first experienced the power of Lean and Agile methods by leading an Agile adoption as Director of Planning and Program Management at TransUnion. Since then, he has steadily grown SoTech Advisors by seeking knowledge of new and better management methods at every opportunity.

Working with enterprise clients to adopt, integrate, and adapt Lean and Agile practices, Mr. Light helps people think and act in ways that deepen their understanding of the work they do. He focuses on pragmatic techniques that increase organizational capacity by improving leadership capability. He can be reached at adam@sotechadvisors.com.



Managing Complexity: Creating Leaders at All Levels

by Esther Derby

I speak to many executives who want faster (and better) decisions throughout their organizations. They want employees to take responsibility. They want innovation, engagement, commitment. The benefits they seek are compelling.

All these executives acknowledge that it is easy to talk about wanting these things, but not so easy to make them happen, in spite of genuine effort, consistent modeling, and good intentions. Why? The reason lives in the way we have learned to think about and structure organizations.

Many of the startups I visit feel alive. They've got a certain energy. People are there because they want to be. They believe in the product; they believe in their chance to make a difference. They're not there just for the sake of a job. They are invested.

But over time, as companies grow, that changes. People may still talk about the vision — and truly believe it — but they don't feel it in the way that the founders and the first 15 employees did.

As more people join the company, the founders hire professional managers, marketing people, an HR person (usually an HR lady). Hierarchy seeps in, along with functional departments, job descriptions, formal policies. It's harder and harder to count on people to do the right thing without someone to guide them.

Sooner than you would imagine, engagement slips away, and people don't seem to get it. They do their

jobs, but the sense of pulling together is gone. It seems that people at the top know what to do, and the people at the bottom are just doing their jobs. This isn't because the people at the bottom are lazy, unaccountable, dull slackers. It is because they don't have the knowledge, means, and boundaries to take the initiative.

KNOWLEDGE IS BIFURCATED

Bifurcation of knowledge is a fact of life in most hierarchies. People at the top understand the context. Founders, the first 15, and key managers know the business, the market, the product, the customers. They hold the financial information about how the company makes money and the current financial status. Since they hold this info, they also know what the company should do — on a strategic and tactical level. Knowledge flows down, but mostly on a "need to know" basis — a trickle, not a torrent.

At the same time, the higher you are in the hierarchy, the more the day-to-day reality of the business is abstracted to numbers. People at lower levels in the hierarchy don't want to displease people at higher levels, so they tend to minimize bad news — and the picture becomes rosier with each level it ascends.

On the other hand, knowledge about how things really work, how to work the (doesn't-work-as-defined) process, and what the status really is gets concentrated at the bottom of the organization. Hierarchy filters and distorts information in both directions. If the two sets of knowledge overlap, it is often in middle management (see Figure 1).

THE ILLS OF BIFURCATION

As companies grow, bifurcation becomes more pronounced and contributes to a number of unhelpful dynamics.

Dependency and Lack of Initiative

The less the "doers" of an organization — the people who make products and deliver services to customers, who

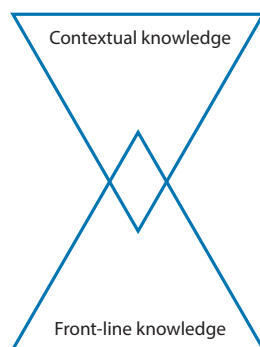


Figure 1 — If contextual and front-line knowledge overlap at all, it is usually among middle managers.

live in the bottom triangle — understand the context, the more they wait to be told what to do. In the absence of contextual knowledge, they may make poor choices.

Erosion of Trust and Respect

When people do not understand the work that other people do, it's easy to not respect that work. People in the upper triangle (managers) may expect or promise things that aren't possible, given actual working conditions. "Doers" often wonder whether managers do real work or simply attend meetings. Managers wonder how something that seems simple to them can possibly take so long.

When I worked in financial services, we had an assignment to program our system for options trading. Compared to straightforward purchase and sales of stocks and bonds, the new functionality was a bit complicated to code in a procedural language. Looking at the estimate (1,000 hours of programming time) and considering the difficulty he'd had wrapping his head around puts and calls, one of the programmer suggested it would be better if we told the portfolio managers not to trade options.

That statement sounded stupid, but the person who made it was far from dull. The problem was that he didn't know much about the domain and didn't understand the part that options played in managing risk. He lacked contextual knowledge.

About the same time, one of the portfolio managers asked us to program the computer so that he could simply make his requested trades verbally. It is true, speaking the trade request would have been much less work. With the then-current system, the portfolio manager had to look up the identifying number for each investment vehicle, write it down, and note the number of shares or instruments to buy or sell, before handing it off to an assistant.

Our trading system ran on an IBM mainframe. Voice recognition was just coming out of the university labs. I understood how attractive a voice-activated system would be, but his request seemed out of touch.

Neither individual understood enough about each other's work and world. This makes people in one triangle appear foolish to people in the other triangle.

Unreasonable Expectation That Leaders Must Always Be Right

This is both unrealistic and a huge burden. When managers don't have the answers and make mistakes, it furthers the cycle of distrust and disrespect.

It's How Hierarchy Happens

All of these dynamics are both cause and effect. Given the typical trajectory, most companies end up with a hierarchy in which people at the top set strategic objectives, people in the middle direct and coordinate activities, and people at the bottom perform activities (see Figure 2). This is the dominant pattern, and bifurcation of knowledge is one of the factors that holds it in place.

BREAKING THE CYCLE

Breaking the cycle requires a shift in how we think about leadership *and* a shift in how we manage. Many of our current management practices evolved from earlier times, when workers may indeed have been uneducated (early factory work, building railroads) or unwilling (yes, on slave plantations). But that is not the case in the modern software organization. The people doing the work are highly educated and very willing ... until bifurcation robs them of necessary contextual knowledge and the opportunity to take meaningful responsibility for their actions and decisions.

Rethinking Leadership

Let's look at leadership first. Many popular definitions of leadership emphasize charisma, vision, or position. The darker definitions hint at manipulation or even coercion. In software companies, though, people throughout the organization are smart, well intentioned, and capable of making good decisions. So we need a different definition if we want to make our companies more flexible and smarter. Here's one from Jerry Weinberg:

Leadership is ... the ability to enhance the environment so that everyone is empowered to contribute creatively to solving problems.

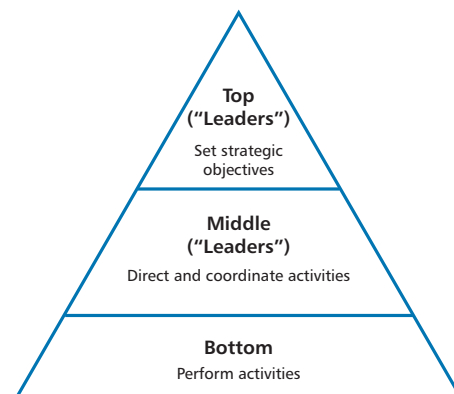


Figure 2 — Bifurcation of knowledge leads to hierarchy and holds it in place.

Under this definition, everyone can take action to improve the situation and solve problems. However, this definition worries people who are accustomed to more traditional models in which people at the top set strategic objectives, people in the middle coordinate work, and people at the bottom merely do. And, in fact, without appropriate information, people will make decisions that are incorrect, wasteful, and harmful.

Organizing Work by Domains

That's where the second mental shift comes in. We need a different way to approach how the organization accomplishes its work and makes decisions in an appropriate time frame. Rather than think about hierarchical levels (which accrete power and privilege), let's think about domains. I call these domains Steering, Enabling, and Doing. These domains don't imply hierarchy, but rather pace of change and granularity. Steering-level concerns (which pertain to the overall directionality of the system) usually change more slowly than those in the Enabling domain or Doing domain (see Figure 3). Each domain can be further described as follows:

- **Steering** sets a course and steers the entire organization. Steering is responsible for the overall viability and functioning of the system. Steering deals with the *raison d'être* of the company: value proposition, market positioning, relationship with customers, difference in the world.
- **Enabling** supports activity that works toward the organization's purpose. Enabling is responsible for dynamic allocation of funds, managing tradeoffs, prioritizing. Enabling attends to the conditions that support people directly involved in creating products and services so they have the means to accomplish work and deliver value. This includes providing those means, articulating boundaries, clarifying decision rights.



Figure 3 — Three organizational domains.

- **Doing** delivers value to customers by creating goods and services and interacting with customers. Individuals in the Doing domain have great discretion in how they organize their work and how they work across teams.

The Three Cs: Factors That Enable Success

Within each domain, three factors allow people to create the conditions that foster leadership at all levels, great work, and great results (see Table 1):

- **Clarity.** People know what to work on. They know how their work fits into the big picture. They know who the customer is and the relationship the company wants to create with both customers and employees. A keen sense of the company's purpose and customers is essential guidance for everyone in the organization.
- **Conditions.** People have the means — budget, facilities, equipment, access to expertise — to do the work. Organizational structures and policies support the work.
- **Constraints.** Constraints that inhibit flow get in the way of work. Constraints that reduce flailing support work. Too many choices will delay decisions. Ambiguity about focus leads to futile arguments. With appropriate constraints, people know what should always be done — and what should never be done. People can articulate their bounded autonomy. They know both the decisions and the parameters for those decisions. For example, a team may have the freedom to choose their own testing tools, with a budget of \$30,000. Bumping into a boundary signals the need for a conversation. Say the team uncovers a testing tool that will save them enormous time but costs \$31,000. This situation triggers a conversation with the person who allocates funds, so the team can make the case for the additional expenditure.

Applying the Three Cs In and Across Domains

Within each domain, the concerns differ, but the overall aim is coherence. Coherence minimizes contradictions between what is claimed and what is done. Coherence allows appropriate local variation based on the nature and needs of the work. For example, accounting groups tend to have tighter constraints about the way they do their work than a product development group. However, both must work toward the same overall purpose: solving problems and creating value for customers.

Steering

Within the Steering domain, clarity relates to the purpose of the organization (product, value proposition),

the design of the business model, market position, and the desired relationship with customers. Decisions about balancing the interests of customers, employees, and stakeholders reside in this domain.

Conditions concern the overall organizational structure (organizing by product line, function, region, network, etc.). Global policies that set the tone for the organization, such as compensation philosophy, are also in the Steering domain.

Heuristics or principles set constraints in the Steering domain. Rather than elaborate policies and procedures that attempt to dictate behavior for all foreseeable situations, heuristics and principles are broader statements that help people act, interact, and make decisions in most situations without being prescriptive.

For example, the heuristic “Company before department, department before team, team before individual” guides decision making in a way that enables people at every level to make reasonable judgments, without having to seek permission and approval. People only need approval when actions bump up against an explicit constraint, such as the \$30,000 tools budget mentioned above.

Steering constraints set out what must be consistent across the organization and what can evolve to meet local needs and conditions. One company I visited set workspace guidelines based on real estate costs at headquarters (New York City). The result was cramped offices for New Yorkers and spacious ones for people in the Upper Midwest. When the New York employees visited Minneapolis, they were not pleased.

Enabling

In the Enabling domain, clarity is about what teams work on, in what order, to meet the company’s purpose. This is where portfolio management, roadmaps, and

release planning live. Clarity also includes articulating compelling goals. Effective goals act as attractors, articulating a purpose, and tapping into pride in work. A compelling goal may address solving a customer problem, overcoming a technical challenge, or improving customer experience.

The key is to focus them on outcomes, rather than activities or targets. “Achieve 90% test coverage” is a target. It’s quite possible to reach that target without improving quality one bit. “Hold code reviews on all changes” puts the focus on activities, without saying why. “Improve our customers’ experience installing our product” is more likely to get people thinking about why the install process is unsatisfactory and how to make it better.

Producing appropriate results requires that people be organized into teams in ways that foster teamwork and flow. How work enters teams and how teams integrate work that crosses team boundaries are conditions for the Enabling domain. Constraints reflect decision rights and bounded autonomy granted to teams.

An additional — and critical — function crosses the Enabling and Doing domains. Problem finding/fixing is essential for system health and requires robust and open communication between people in these domains.

Doing

And it all comes down to Doing. Clarity, conditions, and constraints must work together to create the right balance of freedom and responsibility for teams. Without appropriate freedom, companies waste the experience, intelligence, and creativity of their employees. Without appropriate responsibility, teams may miss the mark or do foolish things. Conditions make it possible for teams to do work. Clarity and constraints bound

Table 1 — The Three Cs in Each Domain

	Clarity	Conditions	Constraints
Steering	<ul style="list-style-type: none">• What is our purpose?• Who are our customers, and what relationship do we have with them?• What problem(s) do we solve for them?	<ul style="list-style-type: none">• What is our overall organizing principle?• What kind of workplace do we want to create?	<ul style="list-style-type: none">• What principles and heuristics guide us?• What must be consistent across the organization, and where can things evolve locally?
Enabling	<ul style="list-style-type: none">• What efforts should we fund and work on?• How do we order our work?• How does day-to-day work connect with our purpose?	<ul style="list-style-type: none">• How can we organize people for effective collaboration?• How can we flow work into teams?	<ul style="list-style-type: none">• Which decisions flow to teams, and which are shared with management?• What constraints balance freedom and responsibility?
Doing	<ul style="list-style-type: none">• How can we best organize our work?• What problems are emerging, and how can we address them?	<ul style="list-style-type: none">• What adjustments do we need to make to our team environment?• How can we best coordinate with other teams?	<ul style="list-style-type: none">• What agreements will support our work?• What approach is most effective now?• What tools and training do we need?

autonomy and maintain the balance between freedom and responsibility.

Unfortunately, in my experience, many teams are either overconstrained or underconstrained, and freedom and responsibility are out of balance. Overconstraint can be direct or indirect. Direct constraints explicitly limit discretion, self-determination, and decision making. I visited one team that was granted permission to decide their team name, their team meeting times (and penalties for showing up late), and how to spend their snack budget. The team's managers made all the significant decisions: what stories to work on, how to implement functionality, task assignments, vacation schedules, tools, training, and so on. In another project, testers were forbidden to talk to developers and customers. Each person was given a weekly task list but not permitted access to the overall plan. The direct constraints on these teams depressed engagement and limited use of people's creativity and problem-solving skills.

Other teams have the appearance of reasonable autonomy but are hampered by organizational rules and standards. These indirect constraints come from formal processes for procurement, long lists of (sometimes contradictory) standards, required documentation, gates, finely grained job descriptions and levels, work rules, and so on. Such constraints can originate in any domain, and many of them seem reasonable when looked at one by one. The problem is, they all cascade to the Doing domain and make it nearly impossible for doers to accomplish the organization's purpose. Taking initiative puts people at risk for censure — but breaking rules is the only way to produce tangible results. True leadership is not part of the equation.

Some very unfortunate teams are overbound with both direct and indirect constraints. All they can do is wait to be told what to do and accomplish what they can as they navigate the maze of restrictions. There's not much hope of tapping into experience, creativity, and intelligence here.

Underconstrained teams slog toward uncertain results, but for a different reason. A while back I was consulted about one such group. After two years and zero tangible results, their manager called me to ask what to do. This team had been arguing about what problem they were trying to solve for two years. Several members held strong and nearly opposite opinions. They argued back and forth, and when it looked like one view would prevail, a third member would interject a different idea. Off they'd go again! The team had neither an appropriate attractor (clarity about the problem and organizational need) nor appropriate constraints regarding decision making and delivery. So, predictably, they were flailing.

Also predictably, the endless (and needless) conflict eroded trust and relationships.

Teams need to be able to do their own adaptive planning, make their own commitments, and organize their own work — and thus have clarity for how they will accomplish the organization's purpose. They can and should adapt and improve their own internal conditions. They should be able to decide on their own approaches, working agreements, and lateral linking. I see no reason why teams can't make boundedly autonomous decisions about hiring, tools, training, and more. When this happens, it's much more likely that innovation, engagement, and commitment will thrive.

ENABLING LEADERSHIP AT ALL LEVELS

Hierarchy isn't evil or bad. I'm not saying it has to go away. In fact, hierarchy can provide essential structure and support for people doing front-line work. But we do need to find different ways to think about and manage organizations. The prevailing model may have worked for certain forms of work, but it works against effective knowledge work.

Rather than decomposing work into functions and tasks, consider Steering your organization, Enabling the flow of work, and balancing freedom and responsibility so the Doing domain can go about the business of building products and creating services. Manage with clarity, conditions, and constraints to unleash productivity and speed.

There is no one right way to apply this model. In fact, competitive advantage rests in adjusting these factors to your context and culture. When you enable leadership at all levels, you will achieve better and faster decision making, greater flexibility, and increased engagement — and vastly reduce the need for close supervision. People might just start to love coming to work again.

Esther Derby is a recognized leader in the human side of software development, including management, systems thinking, organizational change, collaboration, team building, facilitation, and retrospectives. Since 1997, she has run her own consulting firm, Esther Derby Associates, Inc. Ms. Derby helps managers create the conditions for success and works with companies that want to do better at delivering valuable software to customers. She is an expert in team and organizational dynamics and a leading thinker on managers' roles in Agile organizations. Ms. Derby is the author of more than 100 articles, coauthor of Agile Retrospectives: Making Good Teams Great and Behind Closed Doors: Secrets of Great Management, and a former member of the board of directors of the Agile Alliance. Ms. Derby has a master of arts degree in organizational leadership and a certificate in human system dynamics. She can be reached at esther@estherderby.com.



How Lean Management Systems Can Enable Agile at Scale

by Steve Bell and Karen Whitley Bell

Agile practices have achieved widespread adoption in many large organizations; some have reached a state of maturity where their team-level practices are now relatively high-performing, standardized, and replicable. Many of these organizations are now endeavoring to scale Agile to larger projects, programs, portfolios, and products where they are encountering new challenges, often related to the traditional management systems that surround Agile practice.

Fortunately, over the last several decades, Lean enterprises have learned many lessons about managing complex, dynamic organizations across many industries, including manufacturing, financial services, and healthcare, and have developed and documented useful practices. The Agile community can benefit from careful examination of these practices.

In fact, several sophisticated models have emerged for scaling Agile. These include Cutter Senior Consultant Scott Ambler's Disciplined Agile Delivery,¹ Dean Leffingwell's Scaled Agile Framework,² Jeff Sutherland's Scrum at Scale,³ Craig Larman and Bas Vodde's Large-Scale Scrum,⁴ and the organizational model of Spotify,⁵ a Swedish startup. These models are all based, to a large extent, on their respective designers' familiarity with Lean management system principles and practices.

These models share many similarities, in that they propose a multidimensional framework for continuous communication and learning, across organizational functions, product features, and technical components, guided by overall business and technical architecture. Many Agile practitioners find them to be useful logical operating models, but while they all have merit, none should be used as a prescriptive "solution" to the unique circumstances of any enterprise. In our experience, one must have a deep appreciation for a key, but frequently overlooked, Lean principle — the value stream — in order to realize their most transformative potential.

A value stream, in its simplest form, is an "end-to-end" view of a process supported by end-to-end visualization and collaboration. Although simple in its basic

design — like a fractal that starts with a simple algorithm and grows into an elaborate pattern — a network of value streams is not easy to visualize in a complex organization.⁶ In this article, we'll share a value stream perspective on four key software development lifecycle (SDLC) challenges (customer experience, bridging build and run silos, legacy and cloud integration, and plan and control) and discuss two advanced Lean management system practices that promote the ability of the enterprise to foster a value stream perspective and improve overall value and performance.

FOUR VALUE STREAM PERSPECTIVES

Customer Experience

It's not an overstatement to say Lean thinking is all about customer value. Anything that does not create customer value is waste and should be eliminated.

The Agile community clearly believes this as well. Teams focus on customer value by prioritizing stories, continuously testing and evolving the product in rapid cycles, measuring the value, and feeding that information back into future stories and prioritization. User experience has become a distinct discipline, and emphasis on "empathy-based design"⁷ helps teams understand what the customer really wants and feels.

The question "Who is the customer?" always yields interesting insights, and it's a discussion worth having regularly, even within mature teams and enterprises. Customer engagement is so essential to Agile practice, yet often (particularly in large enterprises) there is little understanding of, or empathy for, the end customers. Many times a member of the business team is assigned the role of end customer surrogate (e.g., product owner) to interface with the Agile team. But how well does that business team member understand the real (end) customers? How often does he or she actually talk with customers? How well does this team member really understand the customers' behaviors, needs, and wants?

While much attention is paid to customers, in the end, they are rarely in a position of authority or influence; they are often disintermediated by analysts and disrupted by governance mechanisms. This is especially true in large enterprises, where deeply embedded organizational structures and mindsets can be overpowering. How long have we been hearing about the “customer-centered enterprise” and “delighting the customer,” but with no real change to underlying attitudes and behavior?

The deliberate elimination of waste in all its manifestations through disciplined problem solving is a cornerstone of Lean practice.

There are several Lean lessons to apply here:

- Go to gemba (i.e., visit the front lines where the actual work is done).⁸
- Engage the customer intimately in a highly collaborative, visual setting.
- Work in rapid cycles.
- Measure the impact to the customer with each iteration (i.e., the Plan-Do-Check-Act [PDCA] cycle).
- Let the product evolve dynamically.

When facing organizational resistance, consider creating a single, complete, integrated value stream⁹ that serves as a working prototype to develop capabilities, overcome systemic obstacles, and provide a demonstration case to the rest of the organization.

So once they’ve gone to gemba, it’s time for the team to decide how to approach the challenge of meeting a need. What do the customers really want? Do they necessarily want more technology interventions? No. Of course, every Agile team understands this, but there is an underlying truth within this question. What customers want is a better experience, which may or may not involve technology. Yet when most Agile teams come together, too often they think in terms of how technology can be applied to create a feature or achieve a goal. Rarely do we hear a team approach the challenge by first asking if technology is necessary to achieve the goal, or if, in fact, the process should be simplified and some technology should be removed (with the added benefit of reducing technical debt).

A popular maxim in the Agile community is: “The goal isn’t to write better code faster, but to write less code.” While an improvement over older ways of thinking, this still misses the point. The goal isn’t technology at all.

The goal is customer experience. Technologists must learn to not think like technologists. They need to take a step up and view the customer experience and need at a higher level.

Bridging Build and Run Silos

There is a natural tension between the build and run silos in an IT organization. Build aspires to flow, with smaller and faster cycles, while run has learned to batch releases into large, carefully change-managed events, to create (the perception of) control and stability.

When Lean product development flows through Lean operations, resulting in end-to-end value stream collaboration, we achieve the goal of DevOps — the continuous flow of ideas to delivery and realization of value for the customer. Lean thinkers insist that flow must prevail over batch thinking, so a focus on continuous delivery (aka just-in-time, one-piece flow) improves speed and thus the ability to respond quickly to changing customer needs in a dynamic marketplace.

In order to achieve continuous delivery, an organization must persistently remove the obstacles to speed, which are frequently related to quality. For example, quality deficiencies in the preproduction environment may cause testing to take longer than necessary, requiring several iterations of rework. This creates “failure demand” for the workers, which not only delays the current release, but consumes their capacity that could be directed to value-adding work. The deliberate elimination of waste in all its manifestations (defects and delays are two common ones) through disciplined problem solving is a cornerstone of Lean practice.

Such quality improvement initiatives in IT operations usually begin with a focus on improving incident and problem management processes and developing skilled root-cause analysis and A3 problem-solving capabilities¹⁰ (going beyond the technical causes to the underlying management root causes that allow known technical problems to continue unresolved) in order to measurably improve system reliability, including postrelease incident reduction.

Continuous delivery is naturally simpler to achieve in organizations that rely primarily on modern architectures and tools (e.g., cloud), with the greater challenges typically found in a legacy environment. Here, too, taking a value stream view can create significant gains in overall quality and speed.

Legacy and Cloud Integration

It’s one thing to espouse the virtues of Agile by giving examples of young organizations such as Amazon,

Netflix, Spotify, and others that are built on new architectures and tools. It's entirely another scenario when speaking to older enterprises that must manage significant technical debt, as well as business and technical architectural complexities and interdependencies, often with strict regulatory requirements, across simultaneous legacy and cloud projects. All this adds up to an environment fraught with fragility and risk.

Legacy systems typically handle underlying transactional capabilities, whereas newly developed systems frequently provide customer-facing (Internet and mobile) capabilities, many of which must interact with the legacy transactional systems. It is surprising to many how much mainframe COBOL code still lurks beneath the surface of large enterprises and global supply chains. Hence, many large enterprises are now engaged in costly, multiyear projects to rearchitect and digitize their underlying transactional systems, with varying degrees of success. In any case, a tremendous amount of legacy systems will remain for years to come and must be reckoned with.

The term "bimodal" has recently been introduced to describe a common pattern for treating these two environments, legacy and cloud, as separate IT organizations. The cloud-oriented environment may spring up either as a formal or a shadow IT organization. The legacy development organization retains many traditional waterfall practices, with an emphasis on control and risk, while those working in the new systems — often customer-facing components loosely coupled to back-end transactional systems — develop Agile practices, creating isolated pockets of digital startups within the larger organization.

Though this approach perhaps offers a short-term expedient, the fallacy of thinking bimodally is that these two environments are usually, to some degree, interdependent. In fact, many times they are participants in the same value stream. And remember, in every value stream, the chief constraint to overall speed and quality (the weakest link in the chain) regulates the ultimate output. Fragility and slow mainframe release cycles inevitably affect the value stream and overall enterprise agility. Many enterprises see this as a strategic competitive threat, exposing a flank to aggressive, often niche-focused, market entrants wielding emerging technology.

Cloud systems can be loosely coupled, so their development and release can run on more rapid cycles. But if the transactional foundation of the enterprise must adapt to a changing strategic landscape (as is happening now in financial services), the ability to move toward legacy agility will ultimately be a competitive necessity.

Whatever your impediments, start with the classic Lean inquiry "What is getting in the way of flow?" and start experimenting with how to remove those obstacles.

The only way this can be done is by conducting a high-level value stream analysis across the entire plan-build-run lifecycle, looking for obstacles to flow, and more closely integrating and synchronizing legacy and cloud development, testing, and delivery cycles. While continuous delivery in a legacy mainframe (or change-resistant SAP) environment is a high aspiration, it is clearly possible.¹¹

Whatever your impediments, start with the classic Lean inquiry "What is getting in the way of flow?" and start experimenting with how to remove those obstacles. When the obstacles to quality and speed (from the identification of a need through adoption of value by users) are identified and continuous improvement is underway, you are better prepared to make meaningful gains in the remaining, and perhaps most vexing, segment of the SDLC value stream, which is ...

Plan and Control

Many large enterprises struggle with annual project budgeting cycles; big up-front planning; arduous governance, portfolio, program, and project management and control mechanisms; and other systemic efforts that, while intending to establish more predictability/control and less risk, often do just the opposite. This can cause Agile initiatives to falter or to collapse under the weight of an unrelenting command-and-control mindset.

Burdensome plan and control cause the classic "Water-Scrum-Fall"¹² pattern found in many development organizations (see Figure 1). Water (as in waterfall) represents the up-front planning process, Scrum (or Agile in general) the rapid development cycles, and Fall the traditional stage gates used to regulate release into production. This slow-fast-slow pattern is not flow, yet many organizations with this condition still refer to themselves as "Agile" because of their focus on rapid build cycles.

This deeply embedded command-and-control mindset asserts that if everything can be known at the beginning of the project, then the outcome can be managed and controlled according to plan. It's understandable that when large investments meet with great uncertainty, complexity, and risk, decision makers yearn for a sense of certainty. Nevertheless, there is plenty of evidence

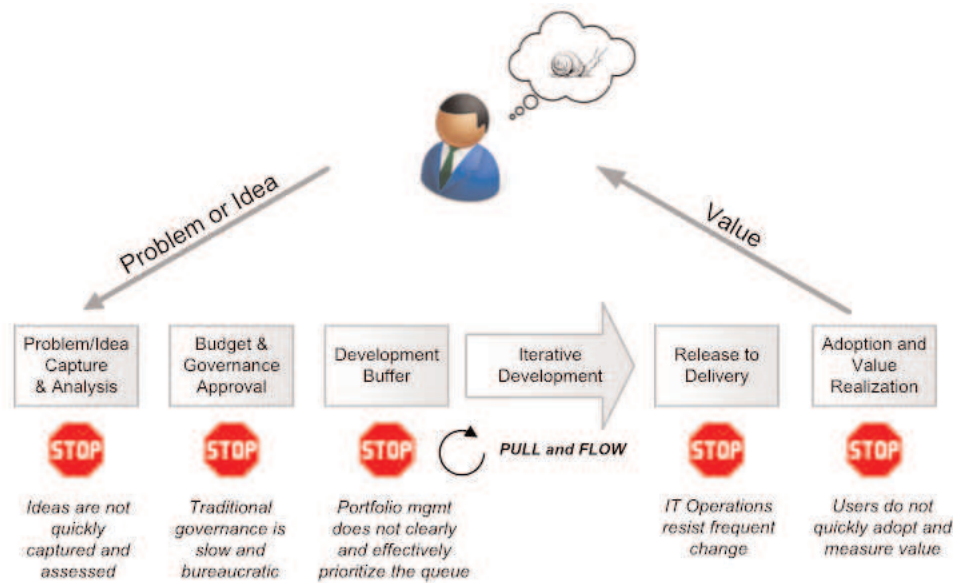


Figure 1 — A common “Agile” value stream pattern. (Source: Bell.)

that emergent learning does not work this way. Success within a complex, dynamic system depends on experimentation, continuous learning, and adaptation.

A broader value stream perspective can help to overcome this mindset. Just as DevOps integrated the build and run silos into a continuous flow, we must also integrate plan into the flow, and eliminate control as much as possible by developing process capability.¹³ By mapping each step of the SDLC, it becomes clear to everyone how many months can go by, and how a significant portion of the budget can be consumed, before the first line of code is written. The result is often hasty, error-prone development and “speed testing” because the project arrives in development already significantly behind schedule and over budget. The snake is eating its own tail. Heavy governance and long budgeting are in response to poor quality and cost overruns, much of which are caused, ironically enough, by the heavy governance and long budget cycles. Once we can see the overall flow, we see the waste. We can then measure it and eliminate it.

In addition to helping to improve overall budgeting and governance concerns, a value stream view will help us address the control orientation of the traditional project management office (PMO) and systemic project management mindsets and practices (e.g., PMI and Prince2). We are seeing many enterprises moving to Agile project management approaches, placing less emphasis on control and more on emergent learning, quality, and flow.¹⁴

TWO LEAN MANAGEMENT SYSTEM PRACTICES

Now that we’ve explored four Lean perspectives on enterprise SDLC value streams, let’s discuss Lean management system principles and practices that can enable and sustain them.

Obeya

Every organization that has achieved some degree of success with Agile has learned the power of visual management and frequent stand-ups at the team level. Some have experience with these techniques at the product, program, or portfolio level as well. It is widely acknowledged in the Lean community that an effective visual management environment promotes teamwork and proactive behavior, reducing wasteful meetings and other communications, while creating a culture of accountability and problem solving. Lean practitioners have learned that visual management is actually a cornerstone of the entire Lean enterprise transformation, as it must extend down and across enterprise-wide value streams and throughout management and leadership systems at all levels.

The Japanese word *obeya* means “big room,” and the classic form is a project room (aka war room), but in fact an obeya can be used to manage any type of work. There is no single prescription for the design of an obeya; it should be purpose-built for whatever work is being managed. An obeya for managing IT operations,¹⁵ for example, may look quite different from one for product development or a transactional process in finance.

What distinguishes an obeya from other forms of visual management is that it visualizes the complete value stream view and illustrates the PDCA cycles within that overall context. To visualize any one portion of the value stream in isolation, and then apply PDCA to that segment, frequently results in suboptimization. An improvement at one point in the value stream may well result in a decline in performance in another. When an obeya provides an end-to-end value stream view, where PDCA (supported by A3 problem solving) and useful KPIs¹⁶ are visualized, teams can see the impact of their actions all the way to the end customer.

An obeya also provides a way to visualize the status of work underway along the value stream and enables teams, as well as managers and leaders, to quickly spot problems. They can then assess and prioritize them based on facts, thus focusing problem-solving attention where it matters most. Once an obeya is in use, it becomes the ideal setting for management meetings at all levels, since the top-level obeya offers a view of the entire situation, with the ability to drill down to lower levels of detail, which may require going to another obeya room on the front lines (i.e., gembu). Consequently, the obeyes often become the most in-demand meeting locations, where the holistic view provides opportunities for learning and collaborative problem solving across teams and projects, thereby reducing waste and fueling innovation.

Many times we're asked, "How can we measure learning?" The obeya — where the flow of work, improvement initiatives, and select current and historical measures of value stream performance (KPIs) are visualized — is where we can find our answer. Learning is demonstrated when we become faster, more capable, and more effective with experience. One approach to gauging learning is to measure and graph over time the improvement of key measures of end customer experience and alignment of improvement efforts and outcomes to overall strategy. These efforts should address the question "Are we getting better at selecting the improvement opportunities that best align to customer value?" Another way of doing this is to measure and graph over time how many improvements are underway or are realized with help (formal or informal collaboration, or even inspiration) from another team. This can show us how well knowledge is spreading horizontally, addressing the question "Are we improving the conditions that enable people to learn from and collaborate with one another?" These measures can help us track our leadership and management performance in the endeavor to foster a learning culture.

Strategy Deployment (Hoshin Kanri)

Imagine what a fractal-like network of interconnected obeyes might look like:

- Starting at the top level of an enterprise, where strategic initiatives are planned
- Cascading down and across the operational levels
- Reaching to and across front-line teams
- Creating a framework for setting targets, developing action plans to achieve these targets, and measuring progress and problems
- Enabling interaction across all organizational dimensions in a disciplined manner

Obeys often become the most in-demand meeting locations, where the holistic view provides opportunities for learning and collaborative problem solving across teams and projects, thereby reducing waste and fueling innovation.

Each obeya is effectively an organizational synapse, where strategic guidance meets collaborative A3 problem solving on a steady PDCA rhythm. Organizations that take this approach learn how to prioritize their initiatives, focusing their efforts on overcoming the obstacles and exploiting the opportunities that yield the greatest value for the entire enterprise. This is the ultimate example of maturation of the Lean management system: strategy deployment — known in Japanese as *hoshin kanri*, which roughly translates as "management compass."

Dan Jones, a founding father of the Lean movement, emphasizes that a Lean management system ultimately focuses the entire organization on a "vital few" underlying problems in order to sustain a transformation. These vital few are often deeply rooted systemic obstacles that make up the DNA of the organization. It usually requires several years of learning by doing for an organization to achieve mature *hoshin kanri*, since many foundational capabilities — team development, value stream analysis, disciplined A3 problem solving, and visual management at all levels — must be in place for it to work properly. But once it achieves maturity, *hoshin kanri* connects the purpose ("true north" strategic goals¹⁷) with the front-line improvement initiatives so the entire

organization can collectively focus on the improvements that will drive the right strategic outcomes.¹⁸

With hoshin kanri, leadership defines top-level strategy, creating a handful of strategic themes and true north targets (such as quality, speed, and customer experience) that cascade down and across the organization as A3 target conditions. Through an iterative “catchball” process, one person states the purpose (tosses the ball), and another responds (through disciplined problem solving, identifying countermeasures that will satisfy the purpose), “tossing” the idea back for another cycle. Catchball is a continuous dialogue, guiding short-term tactical targets at lower levels to overcome obstacles through A3 problem solving, which collectively moves the enterprise directionally toward long-term strategic challenges. Catchball is effectively continuous learning with frequent PDCA cycles at each synapse of the organization (see Figure 2).

When hoshin kanri is adopted in an Agile environment, teams can clearly focus their efforts on what matters most to the enterprise and its customers, align within their value streams, and quickly escalate systemic obstacles that they are unable to overcome on their own. This helps the entire organization to integrate the SDLC with the end-to-end flow of value.

Integral but often overlooked within hoshin kanri (and Lean strategy in general) is the need to align individual and team performance measures and compensation incentives away from siloed behaviors and toward value stream performance and true north strategic goals.

ACHIEVING A LEAN MANAGEMENT SYSTEM

Everyone in a Lean enterprise has three jobs:

1. Do the work.
2. Improve the work.
3. Develop the people.

Doing and improving the work not only includes the productive work of the enterprise, but also the work of management and leadership. In the early stages of Lean evolution, the focus was on operational practices (doing the work), while management systems evolved later. Much the same is happening in the world of Agile, where the focus is now shifting to management systems as Agile scales to the enterprise.

There is no prescriptive checklist for creating a Lean management system — Lean principles and practices must be applied situationally to each challenge. This requires an open mind and a bias toward experimentation. The transformation must be initiated, supported, and sustained by leaders, who demonstrate their intent through their daily actions. They must leave their offices and frequently go to gemba, promoting visual management and coaching A3 problem solving at all levels, practicing humble inquiry, asking “why” with respect, removing obstacles, and learning from and supporting those doing the productive work of the enterprise.

Leaders must abandon the habitual command-and-control mindset and behaviors and venture into uncertainty, emboldened by trust and experimentation. And especially at the highest levels of the organization, where strategy is formulated and guided, leaders must maintain focus on the horizontal flow of value streams, for it is only from their highest vantage point that they can influence the cross-functional mindset and behavior that is necessary to consistently deliver breakthrough customer value.

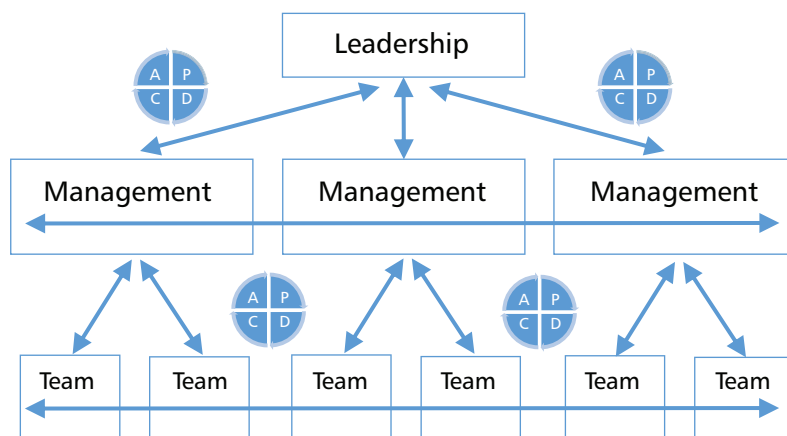


Figure 2 — Catchball.

ACKNOWLEDGMENTS

Special thanks to Jez Humble and Dan Jones for their insights on this topic.

ENDNOTES

¹Ambler, Scott W., and Mark Lines. "Disciplined Agile Delivery (DAD): A Practitioner's Guide to Agile Software Delivery in the Enterprise." Ambysoft (www.ambysoft.com/books/dad.html).

²Scaled Agile Framework (www.scaledagileframework.com).

³Brown, Alex, and Jeff Sutherland. "Scrum at Scale: Go Modular for Greater Success." Presentation to *Agile 2014*, Orlando, Florida, USA, July 2014.

⁴Vodde, Bas. "Introduction to LeSS." *InfoQ*, 30 January 2015.

⁵Lindwall, Kristian. "Leadership @ Spotify." Presentation to *Lean IT Summit 2014*, Paris, France, October 2014; and Kniberg, Henrik, and Anders Ivarsson. "Scaling Agile @ Spotify with Tribes, Squads, Chapters, and Guilds." White paper. Spotify, October 2012.

⁶For a thorough examination of value streams and their application in an IT organization, see Chapters 2 and 3 in: Bell, Steven. *Run Grow Transform: Integrating Business and Lean IT*. Productivity Press, 2012.

⁷Hopkins, Art (ed.) "Empathy-Based Design." *Cutter IT Journal*, Vol. 27, Nos. 6/7, 2014.

⁸Gemba is the Japanese term for the "actual place" where value-creating work occurs. The word is often used to stress that managers must regularly leave their conference rooms and computers behind and go to the front lines to fully understand and support (based on direct observation of current conditions and evidence) the workers as they solve problems.

⁹In Lean terminology, this is called a "model line."

¹⁰The A3 is a one-page problem-solving pattern that can help people solve problems at any level of the organization, leaving behind a storyboard from which others can learn. The A3 approach is based on W. Edwards Deming's Plan-Do-Check-Act (PDCA) cycle of scientific problem solving. To learn more about A3 practice, see: Shook, John. *Managing to Learn: Using the A3 Management Process to Solve Problems, Gain Agreement, Mentor, and Lead*. Lean Enterprise Institute, 2008.

¹¹Kordyback, John. "From Mainframes to Continuous Delivery in 1,000 Easy Steps." Presentation to *FlowCon 2013*, San Francisco, California, USA, November 2013.

¹²West, Dave. "Water-Scrum-Fall Is the Reality of Agile for Most Organizations Today." *Application Development & Delivery Professionals* (blog), 26 July 2011.

¹³W. Edwards Deming once said, "You cannot inspect quality into a product," meaning that most inspection and control effort is waste and should be replaced by efforts to improve process quality and capability so that defects do not occur.

¹⁴Augustine, Sanjiv, and Roland Cuellar. "The Lean-Agile PMO: Using Lean Thinking to Accelerate Agile Project Delivery." *Cutter Consortium Agile Product & Project Management Executive Report*, Vol. 7, No. 10, 2006.

¹⁵Mathijssen, Fred. "Obeya, An Evolution." Presentation to *Lean IT Summit 2014*, Paris, France, October 2014.

¹⁶Our definition of a useful KPI is one that helps teams assess their performance, immediately spot waste and problems, and continuously improve.

¹⁷"True north" must be defined by each organization according to its strategic objectives, strengths, and weaknesses. Common true north goals include customer experience, quality (e.g., fitness for purpose and use, defects released to production, reliability), speed, innovation, employee engagement, learning and capability development, and cost (cost reduction being a natural outcome of the focus on waste reduction).

¹⁸Bell, Appendix C (see 6).

Steve Bell is a faculty member of the Lean Enterprise Institute (www.lean.org) and author of three books: *Lean Enterprise Systems*, *Lean IT* (winner of the 2011 Shingo Research Prize), and *Run Grow Transform*. He previously published the articles "Lean Thinking and Knowledge Work" (with Dan Jones) and "A Leadership Perspective on Lean-Agile Business Intelligence" in *Cutter IT Journal*. He can be reached at steveb@leanitstrategies.com.

Karen Whitley Bell joined Steve as a full-time collaborator and business partner in 2011. Prior to 2011, she worked for 20 years as a hospice and palliative care registered nurse, publishing a book and articles in her professional domain while serving as an uncredited collaborator with Steve on his professional publications. She can be reached at karenw@leanitstrategies.com.



Goal #1: From Activity to Action

by Arne Roock

When a company is small, everything is simple. With just 10 mates in a room doing cool things together, everyone knows what's going on and what's important. They all pull together to make great things happen. There's no need for processes, tools, and rules. If the company continues to be successful, though, more and more people will be hired, and things will start to get more complicated. Some sort of structure is needed, as well as more emphasis on organized communication.

At Jimdo, the website creation/hosting company I work for, we faced this situation when we grew to 50 people. We figured out that we could not work as one big team anymore. So we decided to structure the company into smaller teams. All of the teams were built around their own vision and equipped with everything they needed to continuously move toward this vision. They were staffed cross-functionally, given the freedom to figure out how to best do things, and encouraged to continuously improve their way of working. Of course, it was an investment to get the teams running, and we had (and still have) our ups and downs. But all in all, I would say that these fairly independent, self-directed teams suit us well and helped establish just the right amount of processes and rules we need and feel comfortable with.

None of this is new. Indeed, such an organizational approach is well described in the Lean/Agile literature.

As we grew further, though, we discovered something else that *was* rather new to us. Most (if not all) of the teams were performing well, but they had started moving in slightly different directions, which led to suboptimal outcomes on the company level. What was decreasing was that feeling every great startup experiences: "We are all on the same page, and we can make it happen!" In other words, we were losing company-wide alignment.

Luckily enough, I met Stephen Bungay at exactly that time and read his great book *The Art of Action*.¹ Bungay is a strategy consultant and an historian who focuses on military strategy. One phenomenon he describes is that busy and apparently hard-working companies can come to be dominated by *activity*, with very little real

action. In other words, everyone is doing something, but nothing gets done! What's missing here — and what is needed in order to combine all the activity into action — is clarity. Bungay suggests that we all ask our bosses the Spice Girls Question: "Tell me what you want, what you really, really want!" What is the *one* thing that is, at the moment, most important for the company? There can be only one most important thing, and it is the sacred task of leadership to be very clear on what this thing is, to communicate it to all staff, and to act accordingly when it comes to solving problems or providing teams with resources.

So we sat down together and thought about how this concept could be applied to Jimdo. The result was what we call "Goal #1."

THE MECHANICS OF GOAL #1

How exactly does the concept of Goal #1 work at Jimdo? Each Goal #1 is set by our three founders. They are totally open to suggestions from every employee, but they always have the final call. Before they make their decision, they consult different people, hear different opinions, and share their ideas. The decision is then announced to the whole company in our weekly, all-hands meeting as well as via our internal microblogging tool. Either a new team is formed as the core Goal #1 team, or an existing team is "tagged" as the Goal #1 team. Now everyone in the company has to live by one simple truth: Goal #1 trumps everything! So if the Goal #1 team needs my help, I will help them immediately, no matter which team I am working in. If the Goal #1 team needs management attention, they are going to get it. If the Goal #1 team needs additional people, they will get them. Those are the rules. It's as simple as that.

The next thing that happens is that a feedback loop is put in place. Our Goal #1 Meeting (OK, so it's not a very creative name) usually starts with a three- or four-week cadence and will take place more often as we get closer to the end. The whole Goal #1 team is supposed to be at this meeting. In addition, at least one representative of every Jimdo team (including product teams, marketing teams, support teams, etc.) needs to attend. Anyone in

the company is welcome to attend if he or she likes. The Goal #1 team presents progress (preferably with a live demo) and is open to feedback and questions. After the meeting, the team representatives are responsible for sharing the relevant information with their home teams. Every Goal #1 Meeting is moderated and videotaped, so our remote teams and every individual can watch it afterward. The purpose of the meeting is to align the whole company on achieving the goal by sharing relevant information and allowing for questions. A Goal #1 Meeting usually takes 10-20 minutes.

One or two people are given the responsibility for keeping the communication going, bringing the right people together, being the go-to persons for all kinds of questions and concerns, and for conducting the Goal #1 Meetings. After each Goal #1, we have a big company-wide retrospective to harvest our lessons learned for the next Goal #1, as well as to discuss general improvement opportunities.

WHAT QUALIFIES AS A GOAL #1?

There are several criteria for declaring something a Goal #1:

- It has to be of strategic importance for the company.
- The majority of our teams are involved and need to collaborate in order to make it happen. This amounts to 10-20 teams.
- The estimated time frame for achieving a Goal #1 has been something between three and six months. Now that we are more confident with the mechanics of Goal #1, and having incorporated what we learned from our previous three, we feel ready for shorter Goal #1s.
- Our Goal #1s always include a great deal of uncertainty, usually because we have never done something similar before. For example, our first Goal #1 was to launch a top-notch iOS app for building and maintaining a website. This was a mammoth task for us, because we didn't have any experience with mobile app development. When we decided to build a similar Android app one year later, it did not become a Goal #1, because we already had acquired enough knowledge on this subject to deal with it on a lower profile. What became obvious to us is that the high degree of uncertainty connected to every Goal #1 also means that we can learn lots of things as a company.

Right now we have finished three Goal #1s and are close to starting our fourth one. All previous Goal #1s have been product launches: an iOS app, a redesign of

our platform (including a new style editor), and the integration of Jimdo websites into the 99designs platform. We have had several discussions on whether the Goal #1 concept is limited to product development, and we have always come to the conclusion that it is not. It could also be applied to something related to customer support, sales, marketing, or the like, as our recently announced fourth Goal #1 will be.

What became obvious to us is that the high degree of uncertainty connected to every Goal #1 also means that we can learn lots of things as a company.

WHAT WE HAVE LEARNED

Having seen many different companies in my former career as a consultant, I dare to say that Jimdo is excellent at organizational learning. This is reflected in the way we have dealt with different Goal #1s. No matter how much frustration there was at certain points in time, we took it as an opportunity to learn. Here are a few things we've learned from Goal #1 so far:

Don't Expect Smooth Sailing

For us, Goal #1 is a great tool for creating company-wide alignment and dealing with big and hairy goals. But even after two years, the concept is not a no-brainer. It requires a lot of hard work and includes a great deal of vexation. During our Goal #1s, we have seen great things happening, including unbelievable cross-team collaboration, a unique sense of togetherness, and parties throughout the whole day, including a rate of high fives that was previously unheard of. We've also experienced really deep frustration and demotivation and tearful conflicts connected to Goal #1.

Every Goal #1 Is Unique

If you think about it, it seems quite obvious that every Goal #1 would be unique, but it was hard for us to realize at first, because it destroys the illusion of a blueprint for Goal #1s. Sometimes a dedicated Goal #1 team makes a lot of sense, and sometimes it doesn't. Sometimes a two-week cadence for the Goal #1 Meeting is the best choice, and sometimes it's better to use a one- or three-week cadence. In every Goal #1, different people need to be involved; different roles make sense. Therefore, it's critical to apply certain principles to a new Goal #1 without sticking too much to specific practices. Things might work well in one case but are an

obstacle for the next Goal #1. I would say that not only have our previous Goal #1s been very different, but it's in the nature of the concept that they *have* to be different — and difficult.

Goal #1 is a very powerful tool, so we only use it for the biggest challenges we are facing in the company. As noted, these challenges are unique. If we already had faced the same situation before, we would know how to handle it and therefore wouldn't need to elevate it to Goal #1 status in order to deal with it. As I explained with regard to the Android launch, if we consider something doable without big problems, we save Goal #1 for something else.

Not only have our previous Goal #1s been very different, but it's in the nature of the concept that they *have* to be different — and difficult.

This point also sheds light on why we face so many problems during our Goal #1s. It's easy to believe that these problems occurred *because* of Goal #1 — and I have no way of proving that assumption wrong. However, I truly believe that cause and effect are mixed up here. We did not face problems because of Goal #1. Instead, we only applied Goal #1 to those areas that were really challenging to us, and, of course, it's likely that we would face problems there.

Goal #1 Is Not for Innovation

By applying Goal #1 mechanics, we create radical transparency throughout the whole company. It's like a big spotlight that shines on everyone actively working on Goal #1. On the one hand, that is a good thing, because it creates a positive tension toward finishing things, cutting the scope, triggering critical discussions, allocating resources and staff, and so on. A prerequisite for this is a high-trust culture, where people feel secure that they will not be blamed or punished when failure becomes transparent. On the other hand, it's nothing new that transparency is one of innovation's arch-enemies. Real innovation requires privacy. You need to try crazy things, throw things away, and iterate on the same problem over and over again. This is considered wasteful when the focus is on getting stuff out of the door. Also, innovations always question what we are today. Therefore, we need to be aware of the fact that there will be close to zero innovation as soon as we declare something Goal #1. From that point on, it's all about finishing stuff. There is a context for innovation,

and there is a context for earning money. The way we've designed Goal #1, it's good for monetizing innovation, but the innovation itself has to be created elsewhere.

Radical Transparency Is Hard Work

I remember well when we had a very frustrating period during one of our Goal #1s and someone said: "This Goal #1 is like a bulldozer. We get things done, but it does not feel like fun; it takes a lot of energy." Everyone agreed on this statement, and we discussed possible reasons for it. During this conversation, another metaphor emerged: sometimes applying the Goal #1 mechanics is like pushing an engine into the red zone. That's when it starts feeling tiring and people get frustrated. At the same time, it's a big opportunity, because now we can observe very easily "which parts of our car start shaking first." What we mean by this is that the radical transparency of Goal #1 shows us all the things in our organization that don't work very well — we just need to be prepared to look for them.

To be clear, I am not talking about "low performers" here. The "parts that shake first" are not people; rather, they're processes, policies, and structures. And it's the management's responsibility to find ways to improve them. This was when I realized that Goal #1 can be used for company-wide improvement. It's a great diagnostic tool, but you have to use it wisely, because if you push your organization into the red zone too often or for too long, the damage will be severe!

THE THEORY BEHIND GOAL #1

As described above, when we started with Goal #1, we were hoping to gain more alignment across our multiple teams. One day, when our first Goal #1 was in full swing, one of our colleagues said: "Wow, I did not know our company has so much power!" This was when we knew that alignment had increased significantly.

The origin of Goal #1 was Bungay's work on military strategy and, surprisingly enough, the Prussian army. However, it did not take long before I realized that it's also interesting to look at Goal #1 through a Kanban lens and a systems thinking lens.

From a Kanban perspective, Goal #1 means to limit the work in progress (WIP) for strategic efforts to one initiative.² That does not mean that teams are not allowed to work on other things than Goal #1. It does mean, however, that we treat all other tasks as "intangible" work in the sense that it has to be put aside as soon as Goal #1 tasks have to be done. If you look at it this way, Goal #1

by definition should create slack in the system, as every WIP-limited pull system does. In fact, this is exactly what I realized when teams started complaining that they were losing efficiency because of Goal #1. For example, our marketing teams organize their work as small projects, which usually take a couple of weeks to finish. When a Goal #1 was underway, they were hesitant to start such a project, because they knew they would have to stop it eventually when the Goal #1 made serious progress and needed marketing support. As is always the case with slack, the teams did not like the feeling of being inefficient. One result was more and more requests for estimates and deadlines, because that would give people better predictability and, as a consequence, more efficiency.

I think that we as a company still need to learn to deal with this better and realize that Goal #1 comes with a price. Paradoxically, getting the most important stuff done on a company-wide level can mean getting less stuff done on a team level. This is counterintuitive and feels uncomfortable. But if we learn how to utilize this, it's a huge opportunity, because this slack capacity can and should be used to improve the system.³

From a systems thinking perspective, Goal #1 is an effort for global optimization. According to organizational theorist Russell Ackoff, optimization of the parts of a system will lead to suboptimization of the whole: "If we have a system of improvements that is directed at improving the parts taken separately, you can be absolutely sure that the performance of the whole will not be improved."⁴ And as W. Edwards Deming observed, the inverse is also true: "If the whole is optimized, the components will not be."⁵

What does this mean for Goal #1? As you'll recall, it was our observation that well-functioning teams do not necessarily sum up to optimal outcomes on a system level that led to the introduction of Goal #1 in the first place. When we started optimizing the whole by applying the Goal #1 mechanics, we inevitably suboptimized some of our teams. Most of them could not continue working on their team mission at full steam because they needed to be ready to support Goal #1. Others even "lost" some of their specialists to the Goal #1 team and therefore lost their cross-functionality, which in turn made it very hard for them to complete critical tasks for their team mission. This, of course, led to frustration.

As a matter of fact, not everyone at Jimdo likes the concept of Goal #1 because it heavily interferes with the teams' autonomy. We certainly can improve the way we handle this interference and decrease dissatisfaction, but

in the end it might be a tradeoff we have to make if we want to optimize the whole. As Bungay puts it: "Most organizations suffer from the parts seeking to optimize themselves, which suboptimizes the whole. Taking a holistic view and actually running an organization as an organism takes some courage."⁶

PROGRESS ... WITH A PRICE

With Goal #1, we've found an approach that helps us deal with specific challenges within the company. It's not perfect, and it certainly comes with a price. Therefore, we will continue to develop the format even further. Many of the things that I have described in this article will have changed one year from now.

Is Goal #1 applicable to other companies? I believe that the underlying principles are valuable for any company that has grown beyond a certain size. However, the specific practices need to be changed according to the context each company is operating in. Goal #1 at a service provider will look very different from Goal #1 at a product company, and it's certainly very different at a 2,000-person company compared to a 200-person company.

If you start experimenting with some of the things I've described in this article, please get in touch (see below). I am curious to learn more.

ENDNOTES

¹Bungay, Stephen. *The Art of Action: How Leaders Close the Gaps Between Plans, Actions, and Results*. Nicholas Brealey Publishing, 2011.

²For more on Kanban and limiting WIP, see: Anderson, David J. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

³Anderson (see 2).

⁴"If Russ Ackoff Had Given a Ted Talk" (video) (<http://bit.ly/ackoffted>).

⁵Deming, W. Edwards. *The New Economics: For Industry, Government, Education*. 2nd edition. MIT Press, 2000.

⁶Bungay, Stephen. "The New Model Organisation: A Visit at Jimdo." *Stephen Bungay* (blog), 22 September 2014.

Arne Roock works with Jimdo in Germany. He has a great deal of experience with Lean and Kanban in different contexts, such as start-ups, medium-sized companies, and international corporations. He is fascinated by systems thinking and loves to experiment with ideas from Ackoff, Deming, Drucker, and others. In 2012, Dr. Roock was awarded the Brickell Key Award for his outstanding achievements and leadership in the Lean/Kanban community. He is interested in ornithology, swimming, yoga, and Scotch whisky and has a black belt in karate. He can be reached at arne.roock@jimdo.com; Twitter: @arneroock.



Practical Lean Software Development for Microenterprises

by Andrea Janes

MICROENTERPRISES MATTER

Small and medium-sized enterprises (SMEs) are the dominant type of firm in the US and Europe: 99% of all European businesses are SMEs, and of all SMEs, 9 out of 10 are so-called microenterprises (i.e., companies with fewer than 10 employees).¹ According to the US Census Bureau, over 95% of all US firms are microenterprises (counting all firms without employees and all firms with 1 to 9 employees).²

Microenterprises live by their flexibility and their ability to provide products and services that are tailored to particular needs. However, microenterprises that produce software are often unable to perform software quality assurance due to lack of time, resources, and/or skills. Quality-related information can be difficult to collect and interpret, with the result that investment into quality is often put aside in favor of other activities, such as the development of new functionality.

The lack of time, resources, and/or skills would seem to make microenterprises predestined for Lean software development, which translates Lean principles into the software development domain. Mary and Tom Poppendieck outline the following Lean software development principles:³

1. **Eliminate waste.** Remove every activity that does not produce value for the customer. This can be achieved by just walking around and studying what people are doing or, more systematically, by measuring the time spent for various steps and then discussing if those steps are really needed to create what the customer is paying for.
2. **Build quality in.** Design the process and the product in such a way that errors cannot occur. This principle can be adopted in many different areas. For example, by using test-first development, the development team can ensure that there is no production code that is not covered by test cases. Alternatively, by defining and reusing standard components, the development

team can solve frequent problems in an agreed-upon way that satisfies quality requirements.

3. **Create knowledge.** Design the process and the product in such a way that the development team obtains the necessary data to learn how to improve. This principle means to do one's best to frequently obtain feedback from the stakeholders of the project so that the development process can adapt the product to the needs it has to cover.
4. **Defer commitment.** Delay decisions to a moment in which they can be taken with a very low risk of being wrong. The idea behind this principle is that the time needed to develop a piece of code is wasted if the code has to be deleted and rewritten later.
5. **Deliver fast.**⁴ The idea behind delivering fast is to maximize the learning effect coming from the feedback of stakeholders who can use the evolving prototype. This forces the development team to establish a development and deployment process that copes with changes and maximizes quality.
6. **Respect people.** This principle underlines the importance of developing the skills and technical expertise of people and giving them the necessary power so that they can self-organize to meet the goals.
7. **Optimize the whole.** The idea of this principle is to seek an overall optimum, not a local one. The goal is to ask "What is stopping us from delivering one month earlier?" instead of asking "How can we improve our user interface?" Improving the user interface is not necessarily a bad idea, but it should be valued by the customer — it should be needed. Otherwise, it is wasted time and money.

As valuable as these principles are for beginning with Lean, however, they are not detailed enough to instill Lean *practically*. That requires a constant and concrete analysis of the process to produce value and eliminate waste. Moreover, they are not tailored for microenterprises, which means they ignore the vast majority of the organizations.

PRACTICAL LEAN SOFTWARE DEVELOPMENT

In a book published last year, titled *Lean Software Development in Action*,⁵ Cutter Senior Consultant Giancarlo Succi and I presented an approach to Lean based on company-wide measurement. The key idea in our proposed approach is to log the activities that developers perform (which feature they develop, which part of the user interface they are changing, which part of the database is being modified, etc.) together with the time they spend doing it. In addition to this *process data*, we propose to measure *product data* in order to understand the quality of the obtained product. The distinguishing concept in our approach is that all data is collected *automatically*, except data that is already collected as part of the normal work within the development team (writing documentation, code, managing requirements, etc.).

Data Yields Understanding

The goal is not to spy on what developers are doing, but — together with the entire team — to determine how time and effort are spent and to compare these with the value they provide. The collected data can be aggregated and anonymized; what counts is understanding where the development team can improve. The collected data about the activities the development team performs and the quality of the product can be used to support the above-mentioned principles:

1. **Eliminate waste.** The measured activity stream can be used to perform value stream mapping. By relating how the development team spends their time to the value it has for the customer, the team can see whether what they are doing is creating value.
2. **Build quality in.** Activities performed over and over may indicate that the process or the product should be changed, either to avoid the repetitive activities or to ensure that no mistakes can be made (e.g., though standardization). Also, classes of defects that appear frequently in the bug-tracking system can be a sign that a mechanism (e.g., a test case or a system for continuous integration) should be put in place to make sure such defects do not get deployed.
3. **Create knowledge.** With the client's consent, the logging of activities can be extended to the client side, thereby enabling the collection of data about their usage of the product. The collected data can be used to figure out why certain problems exist, why customers stop using a given product, and how much

value they give to certain features of an application. In addition, the usage intensity can be compared with the development effort to determine whether the time invested in a feature paid off.

4. **Defer commitment.** The activity stream can reveal if certain activities had to be repeated or if certain artifacts (code, documentation, test cases, etc.) had to be modified over and over again. Such activity patterns might be a sign that the development team commits too early to a particular solution before it is clear what the client really needs.
5. **Deliver fast.** The activity stream can reveal how long it really takes to deploy a given application, information that can help the team figure out how to shorten this time.
6. **Respect people.** The collected data about the effort to perform activities can be used to show where the development team has to improve and which skills need to be learned.
7. **Optimize the whole.** Having a log of all the performed activities helps the development team understand *why* their performance is at a given level and find ways to improve it.

The goal is not to spy on what developers are doing, but to determine how time and effort are spent and to compare these with the value they provide.

Our proposed approach is not unproblematic: measuring all activities seems to contradict principle 6, to respect people. Logging all activities might be interpreted as mistrust. That is why I have only seen this approach work in organizations where developers had absolute trust in management that this data would be used in their interest, to improve their abilities and their success.

To respect people, it is also necessary to explain the goals of the data logging, discuss its usage, and give people the freedom to adapt it — and to decide whether to use it or not. I have found that it is useful to begin measuring activities only for a limited amount of time and then to discuss the results. As long as the results are helpful and the benefits outweigh the costs, acceptance will be high.

Tailoring Lean Software Development for Microenterprises

While *Lean Software Development in Action* describes how to practically implement Lean in a company-wide way, it is not tailored to microenterprises, which is the goal of this article. In particular, there are two areas in which our approach needs to be modified to address the special needs of microenterprises:

1. Data collection team. In our book, we recommend the concept of the Experience Factory⁶ for collecting and organizing knowledge. In an Experience Factory context, a separate organizational unit collects and packages experience for easy reuse. This approach, however, is not feasible in microenterprises. Since microenterprises have limited resources, there are no additional personnel to extract knowledge and package it so that it can be used by the developers. Rather, following the advice of the Agile Manifesto,⁷ the team itself has to reflect at regular intervals on which experience is worth collecting and packaging so that it can be reused later.

Data should not be collected “because it is there” but because it supports organizational goals.

2. Definition of organizational goals and strategy. Because of their limited resources, but also because flexibility is the key success factor of microenterprises, it is not feasible for microenterprises to formally define their organizational goals, the strategies they employ, and the management data they need to collect to control the organization effectively. Such formal definition is necessary in large enterprises, as it is an instrument for agreeing on and communicating the organizational strategy. In microenterprises, where only a few employees have to agree on the strategy, informal and less elaborate methods can be used to define goals and strategy, such as a means-end hierarchy.⁸ In a means-end hierarchy, an organization states how (means) it wants to achieve its objectives (ends). For example, an organization might want to increase revenue (end), so it decides to try to acquire new customers (means). The decision to acquire new customers can also be seen as an end, where the means to achieve the end might be to add new features to the organization’s product.

These may seem like small changes, but they have a large impact. In my experience, only a few microenterprises manage experience or define their goals. This leads to situations in which they commit the same mistakes repeatedly. Without clear organizational goals and strategies and a workable approach to managing the organization based on data, the microenterprise does not have the instruments at hand to pursue a clear direction toward its goals. By tailoring the two most cost-intensive aspects of our approach, it is still possible for a microenterprise to implement Lean in software development (see Figure 1).

In the top left corner of Figure 1, software development is shown as a set of *activities*, which employ *resources*, and produce *artifacts*. To collect data about the process, *measurement* uses *automatic measurement* techniques.

To automatically (i.e., without any developer intervention) measure the process, the resources it uses, and the products it creates, it is possible to develop one’s own tools (e.g., plug-ins for the development environment that the development team is using), rely on open source products, or use data already available because it is a byproduct of another activity. For example, because it is the task of source code repositories to maintain the history of the source code as it evolves, they already contain a large amount of data that can be used to understand the time and effort invested in the code.

Interpreting the Data

Data should not be collected “because it is there” but because it supports organizational goals. Even though data is collected automatically, it still has to be analyzed and interpreted to support the Lean principles described above. This can occur in occasional meetings where one collaborator (preferably not always the same person) extracts the collected data from the database and prepares a presentation for the others in which he or she discusses the interpretation of the data, the findings, and suggestions on how to proceed. This approach has a high learning effect: it helps the development team learn how to interpret the data, identify root causes of problems, and internalize decisions on how to improve.

If that approach is not feasible, a second approach to interpreting the data is to analyze it automatically: the development team can decide on predefined metrics that will be used to assess how the organization is progressing toward Lean. These predefined metrics can be visualized in a dashboard to inform the development team continuously about the collected data. A dashboard has the advantage that one can *embed knowledge*

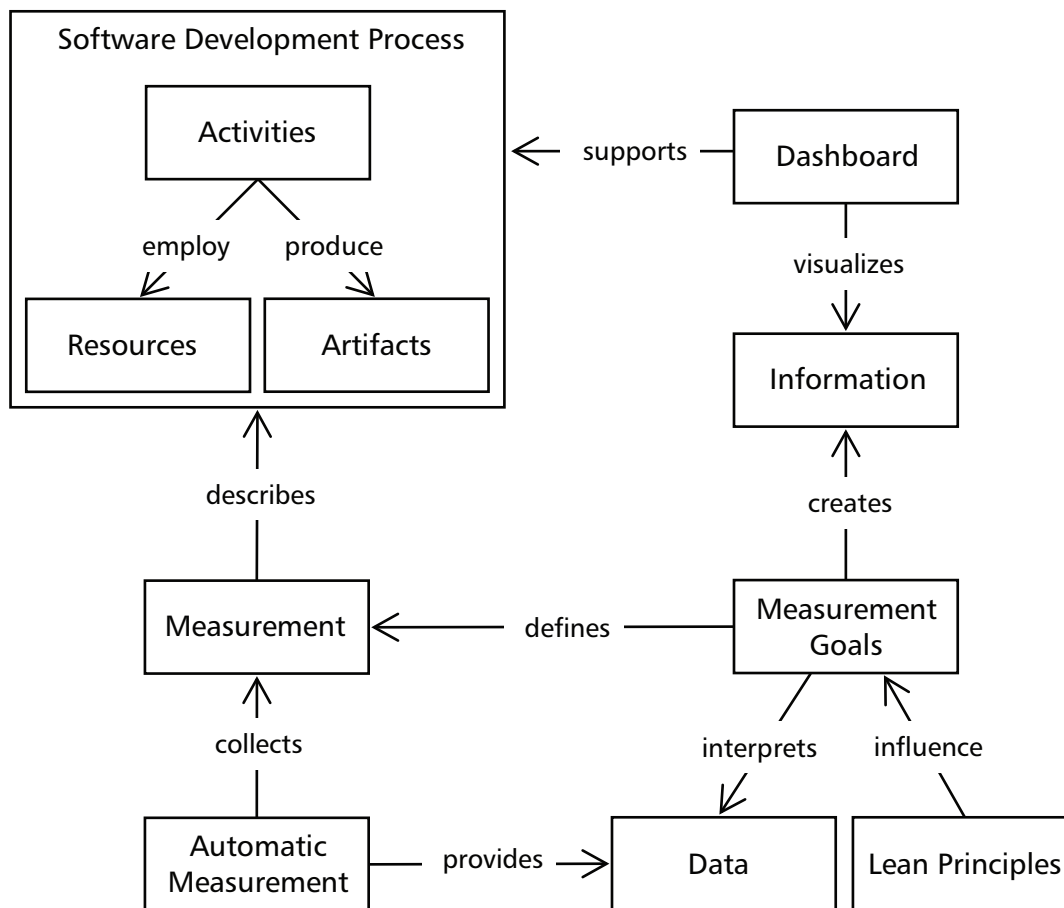


Figure 1 — Concept diagram of instilling *practical* Lean software development within microenterprises.

into a visualization. For example, if the team is convinced that rework is undesirable, they could visualize the amount of rework, measured as the amount of code that is modified shortly after being released, so they can discuss it when it occurs.

Using the second approach, the automatic analysis effort is limited to the development of the dashboard; the discussion of the analysis results and the development of strategies to react appropriately remain tasks that the development team cannot avoid.

Predefined Metrics for Automated Analysis

Below are examples of such predefined metrics together with the Lean principle they support:

1. Eliminate waste. The *time spent per class/namespace/feature/module* helps the team see how much value is created. Time spent coding features that the customer does not need should be discussed, and the development team should look for ways to stop spending time on such artifacts.

2. Build quality in. Using strategies like automated testing and continuous integration, it is possible to ensure stable quality of the products. The *number and type of defects that are still not resolved* are indicators of the progress in this direction.

3. Create knowledge. The *most used feature or set of features* helps the team determine which features should be given priority in the future. By analyzing the source code using open source tools (e.g., SonarQube), other knowledge can be extracted, such as the *complexity* or the *amount of code that is copy/pasted*. Such source code metrics can reveal the quality of the produced source code and help the team evaluate the long-term consequences of the coding methods being used.

4. Defer commitment. The *list of features that were modified often*, the *time spent on those features*, the *classification of those features* (e.g., user interface, database, algorithm) help to identify features or classes of features where either more information is needed

before committing to a course of action or where commitment should be deliberately delayed until more information is available. This does not mean that development stops in the meantime; the developers can create a layer that hides the interaction with the unknown component and exposes an API to interact with it. Later, when the details are known, it is enough to implement the functions behind the API.

5. **Deliver fast.** The *time difference between deployment and the first time a user uses a feature* can measure how well this principle is being implemented. Another measurement that shows this is the *time difference between adding a requirement and its deployment*. It is also interesting to track the reasons *why it is not possible to deliver at a given point in time* and work to challenge — and remove — the impediments.⁹
6. **Respect people.** The *time spent per feature type* shows which types of features create the most difficulties for the development team and where a new strategy should be developed to improve. Repetitive, mechanical activities that “insult the intelligence” of developers should also be automated. The *time spent on repetitive activities* can indicate how severe the problem is.
7. **Optimize the whole.** Measurements that look at the overall performance, such as *the time a feature needs to be implemented from requirement to being used* or *the time lost for functionalities that could not be used because they were not working*, offer insight into how the entire software development process works.

The answer lies in not forcing expensive, creative, knowledgeable developers to fill out time sheets and click on time-tracking systems, but to develop tools that collect the data — maybe not as precisely, but completely automatically.

DATA COLLECTION IS NECESSARY; NOW, MAKE IT EASY

As is evident from the examples I give above, the cornerstone of the approach described here is to measure the time needed to perform development activities. In Lean terms, it is crucial to perform value stream mapping so the organization can understand where value is created along the production steps.

Microenterprises need automated approaches to implement Lean: to collect data, to interpret it, and to be informed about problems. An automated approach does not mean that there is no effort involved; after all, it takes time to develop or acquire the tools to collect the data, to set up tools that help the team automatically test and deploy an application, and to implement tools that measure the produced source code to identify quality issues.

Unfortunately, I observe that many microenterprises opt *not* to invest in automated measurement approaches because they have the impression that they are not worth the cost and effort. Instead, they begin with manual approaches, but having to deal with day-to-day business pressures means they often end up not measuring anymore. The result is that they do not become Lean at all. In fact, I see many microenterprises that struggle to survive because they are inefficient. Some deploy software without thoroughly testing it (thus producing “banana software,” which is software that matures on the client side). Many of the companies develop software using a strict waterfall approach and are frustrated with the results, while young, enthusiastic developers leave microenterprises because it is boring to work there.

The answer lies in not forcing expensive, creative, knowledgeable developers to fill out time sheets and click on time-tracking systems, but to develop tools that collect the data — maybe not as precisely, but completely automatically. It is not technically difficult to write a plug-in for Eclipse or Visual Studio that tracks the time spent per class/namespace/feature/module; an experienced programmer can do that in a matter of days. It also takes just days to interface to source code repositories, issue-tracking systems, continuous integration systems, or source code analysis tools such as SonarQube to extract data about the product or about the results of a quality analysis.

With the streamlined approach presented in this article, microenterprises *can* develop a strategy to become more Lean. Not all aspects of Lean software development can be tackled using this approach, but it gives them a good start for a relatively low investment.

ENDNOTES

¹Muller, Patrice, et al. “A Partial and Fragile Recovery: Annual Report on European SMEs 2013/2014, Final Report.” European Commission, July 2014.

²"Statistics About Business Size (including Small Business)." US Census Bureau (www.census.gov/econ/smallbus.html).

³Poppendieck, Mary, and Tom Poppendieck. *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley Professional, 2006.

⁴The principle of delivering fast might seem to contradict the principle of deferring commitment. The idea in the latter case is to defer those decisions that are costly to change to the moment at which the team members are sure that they can make the right decision. Choosing which database technology to use is typically such a decision, as changing the database later can be costly if all the code that interacts with the database has to be rewritten. Moreover, delivering fast does not mean delivering *all* the expected functionality fast, but rather enough functionality so that the future user can provide feedback. The two principles can also interact: a team might develop a first version in order to understand which database technology suits the requirements best; after this is clear, the team would then commit to a particular database technology.

⁵Janes, Andrea, and Giancarlo Succi. *Lean Software Development in Action*. Springer, 2014.

⁶Basili, Victor R., Gianluigi Caldiera, and Dieter H. Rombach. "The Experience Factory." *Encyclopedia of Software Engineering*. John Wiley & Sons, 2002.

⁷Agile Manifesto (agilemanifesto.org).

⁸Simon, Herbert A. "On the Concept of Organizational Goal." *Administrative Science Quarterly*, Vol. 9, No. 1, June 1964.

⁹I once worked in a company that was developing an accounting system. When a new client wanted to buy and install the system, our strategy was to do that only on January 1. This way, accounting had just concluded its annual statement of accounts, and it was easy to introduce a new accounting system using the numbers of the year-end closing as a starting point for the new year. The work of changing an accounting system during the year was considered more burdensome than making a client wait for months on end to install the new system. Unfortunately, nobody in the company was challenging this claim, and nobody was trying to find a solution. The result is that this microenterprise is still working in this way, losing many opportunities and clients.

Andrea Janes is a researcher at the Free University of Bolzano-Bozen, Italy. His research interests include Lean software development, value-based software engineering, and empirical software engineering. Dr. Janes holds a PhD in informatics from the University of Klagenfurt, Austria. He can be reached at ajanes@unibz.it.

About Cutter Consortium

Cutter Consortium is a truly unique IT advisory firm, comprising a group of more than 100 internationally recognized experts who have come together to offer content, consulting, and training to our clients. These experts are committed to delivering top-level, critical, and objective advice. They have done, and are doing, groundbreaking work in organizations worldwide, helping companies deal with issues in the core areas of software development and Agile project management, enterprise architecture, business technology trends and strategies, enterprise risk management, metrics, and sourcing.

Cutter offers a different value proposition than other IT research firms: We give you Access to the Experts. You get practitioners' points of view, derived from hands-on experience with the same critical issues you are facing, not the perspective of a desk-bound analyst who can only make predictions and observations on what's happening in the marketplace. With Cutter Consortium, you get the best practices and lessons learned from the world's leading experts, experts who are implementing these techniques at companies like yours right now.

Cutter's clients are able to tap into its expertise in a variety of formats, including content via online advisory services and journals, mentoring, workshops, training, and consulting. And by customizing our information products and training/consulting services, you get the solutions you need, while staying within your budget.

Cutter Consortium's philosophy is that there is no single right solution for all enterprises, or all departments within one enterprise, or even all projects within a department. Cutter believes that the complexity of the business technology issues confronting corporations today demands multiple detailed perspectives from which a company can view its opportunities and risks in order to make the right strategic and tactical decisions. The simplistic pronouncements other analyst firms make do not take into account the unique situation of each organization. This is another reason to present the several sides to each issue: to enable clients to determine the course of action that best fits their unique situation.

For more information, contact Cutter Consortium at +1 781 648 8700 or sales@cutter.com.

The Cutter Business Technology Council

The Cutter Business Technology Council was established by Cutter Consortium to help spot emerging trends in IT, digital technology, and the marketplace. Its members are IT specialists whose ideas have become important building blocks of today's wide-band, digitally connected, global economy. This brain trust includes:

- Rob Austin
- Ron Blitstein
- Tom DeMarco
- Lynne Ellyn
- Israel Gat
- Vince Kellen
- Tim Lister
- Lou Mazzucchelli
- Ken Orr
- Robert D. Scott