Agile Product &
Project Management

Vol. 9, No. 6

Executive Report

CUTTER CONSORTIUM

The Four Pillars of Agile Adoption

by Nancy Van Schooenderwoert

The business landscape is littered with failed agile adoption programs. While spectacular successes for agile pilot projects are common, organization-wide adoptions often suffer one of two mysterious deaths: formerly high-flying agile teams unravel — spontaneously, it seems — and attempts to replicate agile pilot success enterprise-wide cause whole ecospheres of agile projects to die off. Like stinking beached whales, these expired agile adoption programs compel us to ask why.

Access to the Experts

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, innovation, 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 print and online advisory services and journals, mentoring, workshops, training, and consulting. And by customizing our information products, training, and 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.

Expert Consultants

Cutter Consortium products and services are provided by the top thinkers in IT today — a distinguished group of internationally recognized experts committed to providing top-level, critical, objective advice. They create all the written deliverables and perform all the consulting. That's why we say Cutter Consortium gives you *Access to the Experts*.

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

The Agile Product & Project Management Advisory Service Executive Report *is published by the Cutter Consortium, 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA. Client Services: Tel: +1 781 641 9876; Fax: +1 781 648 8707; E-mail: service@cutter.com; Web site: www.cutter.com. Group Publisher: Kara Letourneau, E-mail: kletourneau@cutter.com. Managing Editor: Cindy Swain, E-mail: cswain@cutter.com. ISSN: 1536-2981.*

Rob Austin, Ron Blitstein, Christine Davis, Tom DeMarco, Lynne Ellyn, Jim Highsmith, Tim Lister, Lou Mazzucchelli, Ken Orr, Mark Seiden, Ed Yourdon

Cutter Business Technology Council

The Four Pillars of Agile Adoption

THIS MONTH'S AUTHOR

Nancy Van Schooenderwoert

IN THIS ISSUE

Consider the following examples. An agile[1] software team that had achieved spectacular quality and schedule performance enhancing an insurance product was in deep disarray a year later. Despite training and mentoring from very experienced coaches, as well as mastering key agile skills, the team somehow lost its way.

A major property insurance company launched an ambitious agile adoption program, with top expert help, after an agile pilot project delivered its full scope in less than half the estimated time with less than half the usual quality issues. Dozens of agile teams were created, and they delivered. Although the business desperately needed faster time-to-market performance from IT, within two years the agile adoption program was completely scrapped.

We have a mystery here.

This *Executive Report* investigates the agile disintegration from two vantage points: that of a formerly successful agile team and that of a promising agile adoption program that made three fatal errors. The team level is where managers can detect trouble early and head it off. The larger agile adoption program can set up the environment for team success, but advance planning is key.

PILLARS OF AGILE ADOPTION

Every year a growing number of agile conferences present experience reports of high-quality projects delivered in 30%-50% less time than they would have taken using previous methods. Yet, anecdotal evidence from the agile coach community indicates that a very large percentage of companies fail in their attempt to take agile enterprise-wide. What's going on here?

Often when a company has piloted a few agile projects, managers decide that they can mostly go on as before and just cherry-pick some of the agile practices and tools. The "mostly go on as before" part means conducting product development as a *push* system, rather than as a lean *pull* system.[2] So rather than bother with having a product owner (the person who pulls value

from the development team on behalf of the business), managers let analysts decide via committees what features to have the agile team build. Testing is difficult for realistic data and error-handling scenarios so rather than invest the effort, they settle for superficial testing that is too labor-intensive to be sustained. These are just a couple of the typical compromises that pave the way for failure.

There are actually four change initiatives that must be managed successfully if a business is to sustain and spread the success of agile pilot teams throughout the enterprise. Even more critical is that they must occur *simultaneously*. These four pillars of agile adoption are:

1. Teams must be able to produce bug-free software sustainably.
2. Teams must consist of empowered, engaged people.
3. Workflow to the agile teams must be controlled via a "pull" system.
4. Lean portfolio management must be used to control workflow for the organization.

Unless all four of these change initiatives are running successfully, characteristic problems arise. This report looks at these problems first from the viewpoint of a team and then from the perspective of the organization.

AGILE ENTROPY: TEAM-LEVEL VIEW

Let's start by looking in on the Comet team a year after it completed its agile team training. The Comet team is one of several agile teams that was set up in the course of an agile adoption program at an insurance company (the story relayed here is a composite of events that I and other agile coaches have seen). The Comet team had external help for a total of eight months including Scrum training and additional technical training in the use of agile test frameworks as well as help in cleaning up its build process. Its consultant ScrumMaster was onsite most of the time for the first three months and then present every other week until disengagement when the team felt confident to continue on its own.

Comet Team from Agile Adoption to One Year Later

When the consultants completed their work, the eight members of the Comet team were running three-week iterations. While the team's velocity (i.e., productivity track record) had initially jumped around, it had become stable and was gradually increasing. The team members were holding retrospectives at the end of every iteration and checking a few days into the next iteration to make sure they were really following the new resolutions they made in the retrospective.

Their product owner was 50% allocated to that role and was meeting with stakeholders a week prior to the end of each iteration to finalize stories for the backlog. The team's ongoing mission was to maintain and enhance risk analysis software for use by the underwriters. The work it turned out was the best quality the company had seen. The team members knew that there were still bugs in the legacy code but at least they were not adding many new ones. As they improved their tests, they were cleaning out more and more old bugs.

A year later the consultant ScrumMaster saw Tony, the Comet team's lead developer, at an industry conference and asked him how things were going. Tony said he wished they had never gotten into agile at all. They were no longer doing real iterations, just moving the work along in a more or less continuous stream. They had dropped doing the retrospectives after Joanne, who facilitated them so well, moved on to other work. They continued the retrospectives for awhile, but they were just making a quick list of what worked, didn't work, and what they'd change, without delving into the tougher issues. The biggest problem with the retrospectives was that the things they most needed to change required cooperation from another department that was always overworked. So there seemed no point in making the same useless resolutions over and over. The other department was not responding and wasn't going to. That department was invited to send someone to the retrospective, but it never did.

Dissension Within Team

The biggest letdown in Tony's mind was that agile had opened the door to turning software development into a sweatshop. That's the word he used: "sweatshop." Before the conversion the developers had their own cubicles. In the enthusiasm of early agile they'd given them up in favor of a team room. They were happy for eight months with the team room, why the problem now? They were arguing a lot now, and being together all the time in a team room only increased the tension.

Stressors

They were arguing a lot because of disagreements over how much time to devote to cleaning up quick fixes they'd had to put into the code versus getting new functionality in place that their customers needed. At least that's when the arguing began. Now there were many running arguments on all sorts of things. The quick fixes got rushed into place when new features activated a

couple of latent bugs in the code that their agile test framework didn't yet cover. Those bugs got a set of insurance policies written with incorrect risk assessments, forcing the legal department to sort out a remedy. That brought uncomfortable scrutiny on the project manager who'd been fine with team autonomy so far. But now he was being criticized by his manager for not having been firm enough with the team.

Repercussions on Team, Product Owner, and Managers

The product owner had cut way back on the time he devoted to his role because the team had been delivering so predictably month after month, that is, before the problem with the bugs and the legal department. He felt that at some point agile should just be able to run on autopilot; besides there were many other demands on his time. He was feeling "caught out" when legal had to come to the rescue. At first he defended the team, understanding the team's explanation about the test framework not covering that part of the software yet. But he became dismayed by the team's incessant arguing and no longer wanted to be in the product owner role.

Both the product owner and the project manager were starting to wish for the good old days when you just had the analysts write a specification and hand it over to a virtual team, one where the project manager rotated people in and out of a dispersed team and assigned all work tasks. It was getting easy to forget the headaches associated with that process when these new problems seemed so intractable.

Team velocity was steadily decreasing. So managers pressured the team to speed up. With each iteration, the team responded by promising more and delivering less. Often, nearly half the stories committed to were not completed in the sprint. In an effort to work faster, the team members no longer helped each other out and were working solo on the tasks the individuals claimed for themselves. Everyone missed the teamwork they used to have but didn't know how to get it back again.

Tony was using the conference to renew his contacts elsewhere. He had decided to leave the company.

Issues and Analysis

It is clear that in this agile conversion the team learned how to operate using agile practices — to an extent — but managers did not learn how to monitor this new agile system and use the leverage points that could have kept the situation from spiraling downward. Pleased by the agile turnaround, they expected it to just remain on course of its own accord.

Agile systems don't automatically stay on track. No system does; they all need some degree of monitoring and adjustment to counter entropy. In an organization, the "agile system" consists of the core team, plus the stakeholders[3] who've chartered that team, as well as others who manage or interact with the core team. As part of an agile adoption program, managers need to learn the critical role they must play in fighting entropy by exercising control in a new way.

Let's look at a simple analogy. If enforcement of traffic laws was suspended, it wouldn't be long before there would be chaos on the roads. People who want to follow the laws would have to respond to those who don't, and the whole thing would spiral downward. Likewise, in an organization, managers must understand the laws — the key leverage points — and use them correctly to keep the agile system running with minimal ad hoc intervention.

Let's take it one step further. Suppose there has been a breakdown, as we've seen with the Comet team. What are managers to do? An accident, flooding, or downed trees can create a big traffic jam on a road. Regardless of the cause, the individual motorists are powerless to move their vehicles once they are caught in the gridlock. No amount of pressuring or punishing *them* makes any sense. The first thing the police do is *not* to pull cars out of the jam, but to halt the inflow of more cars. They block off the routes into the gridlock, thus preventing it from growing. Next, they free cars out the other side by clearing the blockage.

The jam is a *system* problem, not a problem of the individual motorists. Although each driver has responsibility for controlling his or her vehicle properly, that won't prevent or solve traffic jams. Likewise, many of the problems that software teams experience are really *system problems* of the company. Executives and managers govern that system and have a special role to play during and after a change to agile.

The Four Change Pillars

What should be done in the case of the Comet team? The set of problems it encountered can be viewed in terms of the four change pillars. For each one, we'll look at the leverage points that could have stopped agile entropy.

Problems with Pillar 1: Sustainable Bug-Free Code

Agile teams must be capable of delivering bug-free software. That doesn't mean zero bugs every time, but it does mean that they should have negligible impact.

Troubleshooting and fixing bugs should not take up a noticeable amount of an agile team's time. Agile technical practices such as continuous integration, refactoring, and automated unit testing are designed to achieve this goal and keep the code base in a fit condition for further work.

Agile technical practices shift the mindset from:

- Specify -> code -> integrate -> then do "infinite" test and fix loop, till resources depleted

To:

- Specify via tests -> code -> integrate continuously, so code is always releasable

Agile teams must be able to estimate their work and — in the absence of external blockages — be able to deliver on their estimates every time. Managers are partners with the teams. They fulfill their part by promptly removing blockages and by providing the team with resources to fully test their code *prior to* each release. (Of course, managers also create the organizational framework for team success, which will be discussed in a later section.)

Managers are partners with the teams. They fulfill their part by promptly removing blockages and by providing the team with resources to fully test their code *prior* to each release.

Technical Debt and Latent Bugs

Let's look at Comet's bug issue. Activation of bugs in the legacy code via new features added to the code base was the initiation point for the unraveling. "Technical debt" is a term covering hidden problems in the code including unnecessary complexity, unclear naming, excessive coupling, inflexible architecture, and latent bugs. Perfectly running software can have these breeding grounds for bugs. Technical debt shows up when you try to make changes to the software. It makes bugs far easier to create, even for careful developers.

All software has technical debt. But code written the agile way with strong unit tests has much less debt and therefore far fewer bugs — typically one or two orders of magnitude fewer. In Comet's case, technical debt was getting removed incrementally. This is normal. Removing it all in one go would mean having to focus on that alone for quite some time without building new features. That would be unacceptable to the project sponsors.

Without their unit tests fully constructed, the possibility of a critical bug getting through does exist. Even if the unit tests were fully constructed, it is always possible to overlook something in the design of the unit tests. Human error cannot be eliminated completely by agile or any other methodology. With that said, agile teams do routinely achieve bug rates on the order of 0.2 defects per function point, or roughly two bugs per month for a team of five people.[4]

Who Owns Bug Risk?

Management in this case needed to own the risk of bugs. If you view the organization as a system for producing software-intensive products and services, then the maintenance and enhancement of that system is the job of managers. Owning the bug risk means understanding that technical debt is a precondition for bugs and that technical debt is not there by accident. It is a property of the system, which can be controlled.

The bug risk can never be taken to 0%, but it can be lowered to any level that one cares to pay for. This is always a cost-benefit decision that is ultimately up to management, not the team. Managers decide how good a development environment to provide. Even if a perfect testing environment is provided, there is no limit to the number of agile unit tests that can be written for a given code base.[5] You can make the tests as fine-grained as you need. But they will require effort (cost) to maintain.

There is a constant tension between how much technical debt you allow into the code versus how much effort you put into your agile testing strategy to combat it. This cannot be managed effectively from outside the team. It requires a deep and constant familiarity with all aspects of the code. Only a cross-functional core team will have the skill and judgment to do this efficiently. This is why testers need to be part of the team and why the team needs autonomy to decide how the work, and the testing, will be done.[6]

Agile technical practices give the team new skills to take the bug risk far closer to zero than it could have done before. Still, it is for managers to own the fact that accidents can happen, and they must be prepared to untangle the resulting traffic jam if an accident does occur. This is what quality management guru W. Edwards Deming named "common cause variation," variation intrinsic to the system that is not due to any mistakes made by those working within the system.

Particularly in software development, insufficient early investment takes a very heavy toll. The problem is that it's hard to say definitively just how much specification detail is enough, how much requirements analysis is

enough, and so on. Agile teams solve this problem by following a simple rule: in every iteration, deliver tested, working code that customers can directly evaluate. That means deliver whole usable features, not mere technical components.

Takeaway Lesson

Every agile team needs to be able to produce bug-free code. To the extent that you have bugs, your project is out of control. The product owner and line managers of the Comet team should have owned the bug risk by defending the decision to pay down the technical debt incrementally. If they had fully explained the risk and its mitigation plan to higher-ups beforehand, the incident with the legal department needn't have been so damaging to the agile adoption program.

Problems with Pillar 2: Empowerment

Partnership between team and stakeholders is fundamental. If one side can never say "no" to the other, then it is not a partnership, it is a master-slave relationship where neither side can ultimately get its needs met. Partnership is what opens up the entire domain knowledge of a whole team and places it fully in service to the organization. Only autonomous teams can take full ownership of a goal, relieving managers of tedious unnecessary control tasks.

Two things are needed for a core team to step up to its role in this partnership:

1. Autonomy over its work, with a practical way to make decisions as a group
2. Mentoring to continuously improve its skills in the technical work itself

Empowerment of agile teams is not optional; it is crucial for the high degree of focus and dedication necessary to produce bug-free code sustainably. Mere process rules that a company may institute cannot make people engage to the degree necessary for producing such high-quality software. Individuals' commitment, creativity, passion, and energy are needed — not just their obedience to rules. No externally imposed discipline is any match for self-discipline. This is the key to agile (and to lean thinking), which is completely missed by so many would-be adopters.[7]

Let's look at how issues with empowerment surfaced for the Comet team. There are many problems stemming from this, and in my coaching experience, that is common; empowerment is widely viewed as a "nicety" that can be skipped or, even worse, as a step toward anarchy.

Erosion of "Soft" Skills as First Danger Signal

The first danger sign was the disappearance of retrospectives. Group learning and team unity of purpose *must* be renewed periodically. This is a very high-leverage activity that solves a great many thorny problems while they are still tiny. The line manager(s) of those on the team should have noted that Joanne was carrying the load for retrospective facilitation and should have recognized the importance of that skill by helping others get trained to do it well also.

Facilitation is only part of the picture; good data is needed too. Teams should experiment in each iteration with ways to improve their work, recording data on each experiment. Data plus an efficient way to make team-based decisions should have prevented the team's arguments from getting out of control. Comet team's first argument was over how much effort to spend cleaning up the quick fixes in the code when customers were clamoring for new features. A wise technical manager knows that new features built on top of quick fixes only create a bigger cleanup job to be done later. If this wasn't being grasped by the product owner, then it was a point where a technical manager could've reinforced the team's message.

Too often good facilitation is dismissed as a "soft skill" that is not as important as the technical skills and something we can do without in a pinch.

Missing Cooperation Between Departments

Failure to act on issues raised in retrospectives is another problem. The team had repeatedly said it needed more seat licenses for the source code configuration management tool so testers and analysts could update the tests they coded, but the IT support group was too swamped with other work to respond. By letting this issue sit, managers sent a message to the team that it wasn't important. This contributed to the team's feeling that retrospectives were pointless. Commitment to the agile process must span departments; it must span the product life from definition through customer delivery. A product champion or product owner role serves this need.

In an agile adoption program, friction between the agile teams and the surrounding organization will occur. The organization evolved to support waterfall teams with handoffs between siloed job functions. It will have to change if the agile teams are to survive. Management's chief job in an agile adoption program is to watch for the friction points and then adapt the organization to the agile teams. Simply being "agile" is not the point;

the agile pilot teams need to have shown a clear and significant financial win.

Misalignment of Accountability and Control: A "Blame" Culture

The organization was trying to hold the project manager responsible for the team's actions. But the team has to be responsible for its actions. There was nothing the project manager could have done to make the team avoid those bugs. When things go wrong, there is often an impulse to place blame. The need to deflect blame is what lies behind much of the overly detailed planning in waterfall projects. Deming placed "drive out fear" as one of the key principles in his 14 points for management[8] with good reason; fear and blame feed each other. The project manager needed support from those sponsoring the agile adoption program. By allowing others to place blame on him, they let fear grow.

In the case of the bugs that occurred, they were unforeseeable since they were latent in the existing code. Probably the only way to have prevented them was if the company had invested in more thorough testing designed to catch a scenario of that type. But it is very common (and justifiable) for companies to decide not to invest in doing that. In that case, the team was operating within a system that had set this trap for it. Once the bugs had repercussions in the legal department, the organization's blame reflexes kicked in. Too bad they didn't have the benefit of a good retrospective to identify these dynamics and find a constructive way to address them.

Failure to Sustain Agile Feedback Mechanisms

Another key leverage point that managers neglected is the extremely vital feedback loop between the team and stakeholders. A strong product owner actively engaged with both the team and the stakeholders is necessary for agile teams. Like well-run retrospectives, this prevents a host of problems that are very difficult to address in firefighting mode. A product owner continually challenges the customers and stakeholders to think in detail about the features they want built.

One coach said that he saw that the product owner's test set lacked coverage for large sections of one particular feature. He asked the product owner why and found it was because he'd had to go back to the stakeholders for clarifications as new issues revealed themselves. Under the old system, the developer would have filled in with assumptions or an analyst might have made a misinterpretation of stakeholder needs. By working these questions as they emerge, the product owner prevents a great many defects. The role takes time — often as a full-time job — but it is a multiplier for the whole team's productivity. Although projects vary in their demands on product owners, it is always wise to invest in keeping that feedback loop healthy. Give them time to perform the role.

The Comet team's product owner was feeling pressure to minimize the time devoted to his role. The executive sponsoring the agile adoption program (the agile champion) should have been making sure that people in key roles were allowed the necessary time. Managers of a product owner won't be willing to cooperate unless they have bought into their own role in the agile adoption program.

A reminder from the agile champion might have been all that was needed to keep the product owner properly engaged, if the right agile sponsorship groundwork had been laid.

In summary, Comet's management needed to do the following to prevent the problems stemming from insufficient empowerment:

- Recognize the importance of facilitation skills and grow those skills.
- Act on issues arising from retrospectives that are outside the team's control.
- Support the project manager when he or she gets inappropriate pressure to direct the team.
- Support the need of product owners for time to perform the duties of their role.

Takeaway Lesson

Without the right empowerment, the goal of sustainable bug-free code cannot be achieved. A properly empowered agile team is the means to generating sustainable bug-free code.

Problems with Pillar 3: Team's Work Stream

An agile team "pulls" work from the product backlog into a timeboxed iteration. After a couple of iterations, it knows how much it can do in a given period; it has established a velocity. A common problem for managers new to agile is that they want the team to do more than the amount its velocity indicates. So they pressure the team to give artificially low estimates, to work long hours, and so on.[9]

Velocity as an Emergent Property

An essential idea of the pull system is that it recognizes that the team cannot control everything that has a bearing on its velocity. The team is part of a bigger, interconnected system. That system can produce bug-free

code at a certain rate, full stop. That's the velocity. It might be stated as so many story points per week. If faster output is needed, then the interconnections between the agile core team and the rest of the system need to be understood and improved to generate a higher velocity.

Velocity is an emergent property of the system. It cannot be directly commanded. If you get 50 story points this week by working 12-hour days, when the usual is 40 points, you are not working sustainably. You're simply robbing future production to get a peak now. Perhaps a tool to do data profiling would result in higher velocity. Tools are an example of an interconnection between the core team and the organizational system. So is training. If the team is given training to help it write better agile unit tests, that could improve velocity in a sustainable way. So could having a good team workspace. Each idea has to pass the sustainability test; if you could do it indefinitely, then it's a valid way to influence velocity.

A very common mistake is to try to drive velocity directly, through pressure.

Diagnosis of Comet's Work Stream Problem

The Comet team's velocity eventually started decreasing. When more coding is done on top of quick fixes, the code base quickly becomes brittle and working with it is very hard. Changes produce new bugs, and they take time to fix, hence a slower speed. An increasing bug rate tempts people to do even more quick fixes leading to a downward spiral.

In frustration over the developing problems, Comet's managers reacted by pressuring the team to get more work done. Direct pressure cannot solve this; it intensifies the problems. The team responded by making overly optimistic "commitments" and by refusing to devote any time to helping each other with tasks. Undoubtedly, that further slowed the team's progress.

The team could not, in a short period of time, clean all the bugs out of the legacy code. It was not within its control to do that; it was part of the landscape with which it had to cope. The team also could not quickly cover all the legacy code with agile unit tests. It had been agreed that this would be done incrementally along with creating new code. Deming said in his 14 points for management that "the bulk of the causes of low quality and low productivity belong to the system and thus lie beyond the power of the work force"; that applies perfectly here.

In this instance, managers needed to recognize that the velocity was slowing because the group had decided to go with using quick fixes, and it was not taking the time to clean these out of the code. The question technical managers needed to ask is "Why are we using an unsustainable technique to keep code production going at this rate?" Was the team simply bowing to inappropriate pressure to keep velocity up? That's the surest way into trouble. Or did the team really believe the quick fixes would be harmless? If so, then the team should have seen that wasn't the case and reversed course. Retrospectives provide a good way for managers to gain early insight into issues like these before they are compounded.

Lessons Learned for Keeping Team's Work Stream Healthy

Agile teams should receive their priorities through the product owner and no one else. Sometimes standards groups, regulatory representatives, or others expect to direct team activities. They are stakeholders in the team's outputs and should work through the product owner. When a team has too many bosses, it is set up to fail. Note that the product owner does not assign tasks to team members. The product owner collaborates with them and is an information resource for them.

Pressure on a team to make optimistic estimates is often nonverbal, and it may come from many sources. Teams *want* to deliver, and when they err it is always on the optimistic side. It doesn't take very much pressure to move them onto shaky ground. Learning to push back in an appropriate way is one of the last skills agile teams acquire, so new agile teams are especially vulnerable to this mistake.

Wise managers will watch for times when a team commits to doing significantly more work than it usually accomplishes and will ensure that the team knows it'll be supported if it needs to push back on demands made on it. A commitment that cannot be fulfilled is of no help to the organization but can become a magnet for blame. Managers must understand that they have a responsibility to regulate the flow of work to agile teams. It is a simple, effective way to help teams keep their credibility.

The worst thing an agile team can do is to erode trust with its customers. Each iteration should reinforce that trust.

Agile teams that form spontaneously "under the radar" are very often undermined because management has not acknowledged any obligation to regulate their workload as a pull system. They are flooded with too much to do, have no time for learning new tools or for maintaining a unit test infrastructure, and can only hold the line for so long.

Takeaway Lesson

If teams have mastered bug-free code and are empowered, they can still be destroyed by the failure of management to regulate the flow of work to them, that is, to allow them to pull the work in at their sustainable pace. Managers at every level must buy into the idea that they must never jam more work into a pipeline than its proven capacity. The good news is there are legitimate ways managers can increase that capacity.

Problems with Pillar 4: Organization's Work Stream

One of the worst things that can happen to a good agile team is to be assigned to a weakly supported project. That will mean blockages don't get removed. People and other resources get taken away when needed by other projects. The team repeatedly fails to deliver on its commitments, and it loses the trust of its customers.

Need Lean Project Portfolio Management

The previous three pillars of agile adoption are covered by the popular agile methodologies, but this issue of governing the work stream at the organizational level takes us over to lean territory. Unless a lean discipline is used to decide what work an organization undertakes, it runs the risk of spreading its energy too thin. Weakly supported projects will thrash and waste resources. A company that regularly completes 40-50 projects a year should not have 500 active projects in its portfolio!

Projects that have a direct business need are easiest to do in an agile manner because there is a strong pull by a customer. It is more difficult to gain sponsorship for necessary infrastructure projects. There is less will to allocate a dedicated team and pay for the tools it will need. There is no easy answer here, but if the project is truly needed, then it is the responsibility of the business to understand what it will accomplish and be a sponsor for it. A bank had its project portfolio made up almost entirely of IT infrastructure projects because over the years the business stakeholders had refused to prioritize those projects. They accumulated at the bottom of the portfolio list to the point where they could wait no longer. Either regulations or aging systems forced them into the active group. Only about 5% of IT's capacity was devoted to projects that delivered business features, resulting in pressure to outsource IT.

We saw the Comet team vying with others to obtain needed tool seat licenses. Contention among projects can cause teams to fail in their commitments through no fault of their own. This illustrates that it's not enough to just look at individual agile teams: you need to understand how groups of agile teams function within an organization.

Agile Habitat

One agile pilot project can be sustained by almost any company. It will place more or less demands on various other departments and infrastructure, but you can cope with this. As the number of agile teams increases, pressure on certain resources (testing environments, product owner attention, team rooms, etc.) reaches a point where something has to give. Either the agile teams will be reined in and forced to make more and more compromises in their roles and practices, or else managers will recognize that they have to make the organization into a better habitat for agile teams.

A perfectly skilled and competent agile team may still fail. It's the same problem as when healthy animals are released into a habitat that is being decimated. They don't have a real chance to survive even though each individual is fit and knows how to survive. The highest-priority projects should go to agile teams. Portfolio managers will have to regularly kill off those projects that cannot deliver or that are vying with agile projects for resources.

Takeaway Lesson

If teams have mastered bug-free code, are empowered, and their work stream is properly controlled, they can still be destroyed by the failure of managers to match the organization's workflow to its sustainable capacity. One more time: Managers *at every level* must buy into the idea that they must never jam more work into a pipeline than its *proven* capacity.

AGILE ENTROPY: ORGANIZATIONAL-LEVEL VIEW

Just as you gain new insight into a person's behavior when you meet his or her family, so it is with teams and organizations. The way the agile adoption program is sponsored and planned and the projects selected as pilots — among many other things — will have a bearing on an agile team's success. Let's look next at agile adoption and the four pillars from a viewpoint above the team level.

Property Holders Insurance: Two Years of Agile Adoption

Property Holders Insurance, Inc. (PHI),[10] a well-established insurance company with more than 20,000 employees started on an agile adoption program in 2005. Ben, a project manager there, had learned enough on his own about Scrum to give it a try. His project was so successful (13 months' work delivered in six!) that the director of product delivery took notice. In time, Ben was put in charge of expanding the process to

other teams. External agile coaches were brought in to help, and the number of agile projects swelled to 40. However, within a year of this ambitious agile rollout, the agile adoption program had been scrapped and no agile teams were operating. Let's examine how the agile adoption program unraveled at this company after such a high degree of confidence in it had existed. Table 1 shows a chronology of key events at PHI.

Unsponsored Agile Pilot with Scrum

Ben's initial agile project from March to September of 2005 had no sponsorship at all for changing to Scrum as its delivery method. As project manager and as someone who had been helping teams use some agile practices "under the radar" for a couple years at PHI, he had enough control to permit and support the practices. He also had a team eager to use them. When the team completed a project to deliver a new type of property insurance for renters in six months when it was estimated to take 13, everyone was amazed. Quality was well above average, too. Numerical defect data is not available, but delivered products stay in system test until all medium and severe defects are removed. This took only nine days for Ben's project when 20 would be closer to expectations for such a project.

Sponsored Pilot with EUP

In the aftermath of Ben's agile success story, a group of technical leads and architects within the product delivery organization started to discuss possible agile approaches for PHI, and being well versed in use cases and the Rational Unified Process (RUP), which has led to the Enterprise Unified Process (EUP), they gravitated to the EUP methodology. Meanwhile, Ben received a promotion to become a manager in an unrelated department, outside of both IT and product delivery. The combination of Ben's agile success and the group of designers now promoting agile techniques for analysis and design led to creation of the time-to-market (TTM) reengineering team by the director of configuration management in January 2006. See Figure 1 for a partial organizational chart.

The TTM reengineering team brought in a small group of external consultants who worked on defining a new process for project teams to use. At this point, sponsorship for agile adoption rested with the director

Table 1 — Chronology of Events in the Agile Adoption Program at Property Holders Insurance, Inc.

March 2005	Ben's first agile project begins; 13 months estimated to completion.
September 2005	Ben's agile project delivers in six months. Needs only nine days of system test versus the usual four weeks to remove all medium and critical defects.
November 2005	Ben promoted to head a department in another part of the company.
January 2006	Time-to-market (TTM) reengineering team formed.
April 2006	Enterprise Unified Process (EUP) consultants engaged to define specific methodology and design training.
August 2006	Ben invited to be mentor for EUP pilot but declined, saying EUP as they were using it was not agile — too close to the old ways. Pilot starts.
October 2006	Pilot EUP team delivered ahead of schedule with acceptable quality — deemed a success.
January 2007	Ben named by senior VP of project delivery to become head of TTM reengineering team. Ben removes EUP and moves team back to Scrum.
February 2007	Executive management exerts strong pressure on all projects to deliver much faster. Key July deliverables announced.
April 2007	Four Scrum teams underway, each with external Scrum coach as adviser.
May 2007	Agile two-day training rolled out to 500 employees; 40 agile teams started, 13 external coaches in total.
June 2007	VP of engineering prevails in power struggle; brings in Randy, a senior agile coach, to bring agile adoption program back on track.
July 2007	40 teams deliver but not unscathed.
December 2007	New CEO named, agile adoption program scaled down.
February 2008	No more agile teams or coaches present.

of configuration management, who reported to the CIO. But was it really agile? Employees who had used Scrum, including Ben, said that they were placing too much emphasis on up-front design, allowing too many iterations to pass without being delivered into production and allowing very long iterations. By August, these concerns also were being felt by the TTM team, and it invited Ben to be a mentor for its work. He declined because he wouldn't have control to ensure proper agile practices were followed.

The TTM reengineering team launched its pilot EUP project in August. It was a 12-week project, and it delivered in October — ahead of schedule and with acceptable quality. It was seen as successful, though would later be criticized.

Mass Rollout with Scrum

Believing that excessive time and expense had gone into producing a modestly successful EUP pilot project, John, the senior VP of project delivery, appointed Ben as head of the TTM reengineering team in January 2007, thus moving agile sponsorship from the IT area over to product delivery. Ben chose to halt the EUP efforts and move back toward the Scrum philosophy, that is, keeping iterations short (two to four weeks) and delivering as many of them as possible all the way through production. He realized he would need to quickly expand the number of agile teams because executive management had shifted to demanding much faster project timelines from the project delivery organization. Aggressive top-level targets for the whole company were set for July — just five months away.

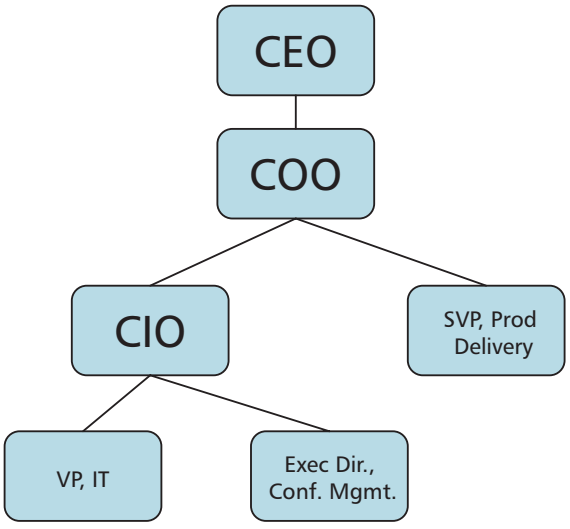

Figure 1 — Senior management organizational chart.

By May, Ben had engaged 13 external agile coaches and was set to roll out 40 new agile teams. The coaches had tailored agile training as a two-day course for PHI, and this was launched to 500 employees. "My biggest mistake was not going slower in the agile rollout," Ben said in hindsight. But the pressure from the executive-management level for July deliverables was tremendous, and there was no way Ben could push back without that being viewed very negatively. It soon became apparent that these agile coaches had different interpretations of what should be allowed, and their varied experience levels began to show. Some coaches told teams that every single "story"[11] must carry customer value, while others said it's OK to have infrastructure stories. These and other instances of variation served to confuse employees.

"We hadn't defined our agile methodology clearly, and this was now hurting us," Ben said. With so many new teams starting and with 13 different external coaches, there was no way Ben could oversee what they were all doing. With roughly three teams per coach, this was a very heavy coaching load. It wouldn't be the case for teams with several months of agile experience, but new teams need close attention.

The teams were able to govern their iterations reasonably well. They tracked their productivity and used past performance to negotiate the content of iterations. They worked very hard to keep the commitments they made. But there was contention for resources, and people were sometimes pulled from agile teams to be used for urgent work elsewhere.

Each agile team was made up of a project manager as team leader, developers, testers, technical lead, and analysts. The project manager was acting in the Scrum product owner role. Software releases were never seen by actual customers until the whole project was completed. This is a compromise in Scrum's methods. Scrum also recommends that teams should be using practices like test-driven development, continuous integration, and automated unit tests, and these new teams were varying greatly in their use of such practices. They hadn't had time to learn those skills in the abbreviated training. As a result, quality suffered. They were producing bugs at a much higher rate than Ben's pilot Scrum team.

Adrian, the VP of IT, now criticized this quality slump and also that of the EUP pilot team. "The product delivery group is not maintaining an acceptable quality standard," he complained. In the matrix organization, IT was chartered with operation and maintenance of products designed and built by the product delivery

organization. Figure 2 shows the relationship between the two organizations. People from IT were allocated to projects run by product delivery.

End Game: Suboptimization Trumps Customers' Needs

The agile teams generally achieved their July deliverables, but quality had suffered. They'd worked long hours and made sacrifices to get the releases out and then were being told that it wasn't enough — more was needed. Morale took a nosedive.

A power struggle developed between Adrian, the VP of IT, and John, the senior VP of product delivery. Against Adrian's criticisms of product quality and too much time wasted on elaborate up-front analysis, John responded, "You can't do projects without analysis, design, and construction. *We* own that." Adrian's view was that agile is only concerned with the production of software — not product management — and so IT should own the agile sponsorship. Ben took this issue all the way to the CEO, confident that he'd agree agile sponsorship should stay with John's organization.

The CEO backed Adrian, and with that, the ownership of agile adoption moved back to the IT organization. In June 2007, Adrian brought in Randy, a senior coach, but not with the aim of coaching teams. Randy's mission was to provide management of the agile adoption program itself and get it back on track. But Randy's expertise in lean and agile gave him the broader view that agile is *not* confined to producing software but is a part of the lean value stream,[12] a chain of steps from identification of a customer need to satisfaction of that need. Randy recommended a complete program of metrics covering the whole value stream, as a foundation to guide agile adoption. But to implement that, Adrian would have to make peace with John and cooperate to gather and monitor the measurements. To Randy's immense frustration, no one disagreed with his recommendations, but Adrian did not move to implement them.

That December a new CEO was named, a change that many said was mainly due to the alarming drop in morale among those who'd been part of the massive agile rollout. The new CEO's message was that we will have steady predictable progress from now on. No surprises. One of the criticisms of agile had been that although time to market was great, predictability was poor. Finishing a project in less than the projected time was viewed as a lack of predictability. The mass of teams started in May would not have reached steady state by July. The earlier teams were pilots. Expecting predictability by that definition was simply unrealistic. Aside from the early stage of the teams, shortcutting the agile technical practices meant that bugs in the code would continue to generate unpredictable amounts of rework.

Within a couple more months, the new CEO had quietly allowed the number of agile teams to dwindle. By February, there were no more of them.

Issues and Analysis

There were quite a few issues in the way this agile adoption program was handled, but three were fatal:

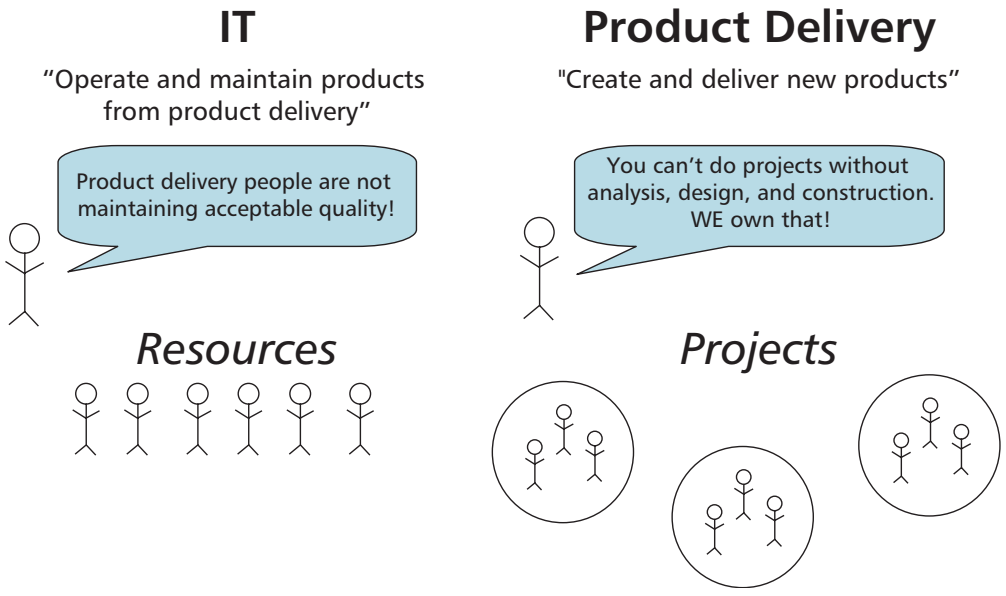

Figure 2 — Conflict between the two groups most affected by agile adoption: IT and product delivery.

1. The fast rollout of new agile teams
2. Compromised agile fundamentals
3. Unclear process leadership

An agile adoption program should proceed at a pace that allows unexpected issues to be handled. A carefully planned pilot, followed by one or two more pilots that take on progressively more complex situations, is the best strategy. There had been 19 months between Ben's Scrum pilot project completion and the main rollout of Scrum teams. During that time, the business vitally needed improved project performance and was getting increasingly impatient for it. Even a very cautious, well-planned agile adoption program would not have let 19 months elapse between initial pilot and the start of serious expansion of agile; on the contrary, it would have seen a fairly steady, predictable expansion. But there wasn't a clear, steady vision behind this agile adoption program. That was a fatal flaw exposed by the rapid rollout.

Agile fundamentals were seriously compromised due to the push to roll it out so massively. Although the Scrum basics can be started rapidly, the engineering skills[13] for delivering bug-free software cannot be mastered in a few days' training. This takes months at best, but, with skilled coaching, teams can start up and learn these skills as they go. This didn't happen at PHI. The agile team members were feeling the pressure, and the atmosphere was one of "do whatever you have to" in order to get work completed — including unprofessional shortcuts. Without the technical know-how to eliminate bugs and the wasteful rework they cause, a vital foundation was missing.

Randy, the senior coach, was surprised by the wide variation in agile methods being taught by the other coaches, and even more by the fact that the teams were free to follow coaches' advice or not: "The coaches were just drive-by advice givers! There was no mandate that teams had to follow any of the agile practices." Few teams had their work tasks up and visible on storyboards. Few had any burndown charts maintained. With such strong pressure from executive levels for the July deadlines, there was a pervasive attitude that anything was fair game if it would get tasks off the "in progress" queue. Unclear process leadership fatally undermined these teams.

Participation by an involved customer (or product manager as customer proxy) could have helped to prevent the situation from becoming a turf battle between the IT and product delivery groups. In this company, customer needs were represented by the project managers as team leaders. From a customer viewpoint, the distinction between who builds the products and who maintains them is unimportant. For agile adoption to succeed, this company needed someone whose authority spanned both groups to advocate for the customer-centric view. The organizational chart in Figure 1 indicates that this individual would be the COO. His involvement had been chiefly to help arbitrate the power struggle between Adrian and John.

Without clear higher ownership of the agile adoption program it became the object of a tug-of-war between IT and product delivery. This is a pattern that has occurred in other agile adoption programs. Agile methods address aspects of software development and also of project management. One of Deming's 14 points for management — break down barriers between departments — is apropos here.

The Four Change Pillars

The four pillars concept can help us analyze this agile adoption program at the middle-management level. Each of the pillars calls for actions by teams with certain support from their wider organization. Let's look at how well the organization played its role in each of the four areas.

Problems with Pillar 1: Sustainable Bug-Free Code

Ben's first Scrum pilot team consisted of enthusiasts who knew or were eager to learn on their own the technical agile practices for producing bug-free software. But the 40 teams in the big Scrum rollout had only an abbreviated two-day training course. The transition from pilot to full rollout calls for more extensive training, not less. The employees making up the bulk of the rollout teams were not agile enthusiasts who'd already picked up some or most of the technical skills of agile development.

The organization took on a serious risk by expecting minimally trained teams to perform as well as the agile pilot team members. This is a lesson that the agile adoption project needed to learn from the pilot teams and carefully apply to later teams.

Takeaway Lesson

Agile software teams must be fully prepared to deliver bug-free software.[14] This demands skills that can be acquired through formal training, mentorship, or individual study. It also often demands more or different access to tools and equipment, because the teams prove the code's quality via testing as they are developing the software. This testing needs to be owned by the team

and tightly integrated into its daily work, not done afterward by another group. Until this capability is in place, other agile practices (burndown charts, daily stand-up meetings, having a product owner) will have limited benefit.

Managers must note the needs of the pilot team for additional test equipment, for access to each other's tools by developers and testers, and for high-touch communication with neighboring groups like production support. They need to consider what the demands on the organization will be with 10, 20, or more agile teams and pace the expansion so that the benefits clearly justify the costs. Avoid expanding in big steps. It gives too many opportunities for unforeseen problems.

Problems with Pillar 2: Empowerment

The company's established practice of having project managers lead and direct the teams was never revised throughout this agile adoption. Teams were free to be more self-directing if their project manager was amenable to it. Laissez-faire is not the same as process change! Accordingly, the level of team autonomy was one more thing that varied too much from team to team. The teams in the large Scrum rollout were under tremendous time pressure — a situation not conducive to getting used to a new way of doing things.

Takeaway Lesson

Empowerment to the right degree is an energy source for teams. It boosts morale and commitment to the project goals. Issues standing in the way of team autonomy, or empowerment, will vary greatly across companies, but the pilot teams are the place to explore this question. They need to be set up in projects that are important but not under severe pressure. Figuring out how to balance the technical know-how of the first pillar with the energy source that results from autonomy done right is the job of the agile adoption project with respect to agile pilot projects. PHI failed to even attempt this element of agile change.

Problems with Pillar 3: Team's Work Stream

PHI's teams did reasonably well in controlling their work stream using agile stories, their estimates, and their velocity to control the amount of work they'd commit to in each iteration. But they were factoring in the drag of having to do rework. Because they were not using the agile technical practices to generate bug-free software, their productivity wasn't as high as it could have been. This helped to make the agile adoption program more vulnerable to criticism.

Coping with rework in this way is feasible so long as the amounts of rework do not vary too widely. There is no way to ensure a constant rework level. Bugs are not predictable. The agile strategy is simply to prevent all bugs. If you "pay as you go" by finding and fixing bugs before software leaves the team's hands, then your measured productivity is a reliable predictor of future productivity.

Takeaway Lesson

Giving teams the control to pull work in (to their iterations) at their sustainable capacity is a healthy way to balance the team's technical estimating ability with the needs of stakeholders for completed high-quality outputs. But there are two important underpinnings to this. One is the aforementioned need to produce bug-free software. The second is the source of agile requirements (called "stories") coming to the team. If story writing is left to the project manager and analysts with no involvement by actual customers, the team could be building the wrong product.

At PHI, a problem that might have been waiting had the agile adoption program continued is this one: Upon product deployment it could have become clear that the requirements the developers implemented were not what real customers wanted. Finger-pointing between the analysts and IT would ensue. Software managers and project managers engaged in agile development are well advised to insist that there be a product owner, product champion, or equivalent person to be accountable for stories, their prioritization, and fielding the team's questions.

Problems with Pillar 4: Organization's Work Stream

PHI started a big push for faster project delivery at the same time as undertaking a very large Scrum rollout. Resource clashes were inevitable, and they hurt agile teams disproportionately. Because agile teams are 100% dedicated to their projects, it is vital that there be no gaps in the resources (tool access, answers to questions, availability of team space, etc.) they need to keep their work flowing.

Agile teams are more sensitive to personnel changes than traditional teams. Because they leverage each person's tacit knowledge by keeping focus on just the one project, it is disruptive for them to lose a team member even temporarily to other projects. Changes in team makeup undercut the team's commitments to its customers, as they are based on the velocity the team has achieved thus far. Several PHI teams had members pulled off to other projects, which no doubt fueled the morale problem.

Takeaway Lesson

An agile team is like a race car with a jet engine on it — capable of great things, but you don't want to be paying for it to sit at stoplights. Eliminate the stoplights by ensuring that everything teams need to run at top speed is flowing to them in streams balanced to the pull they exert.

Lean portfolio management[15] offers a strategy that eliminates wasteful thrashing of projects in the portfolio, making it less necessary to shift people on and off projects. Once an organization moves beyond the piloting stage and starts expanding the number of agile teams, it will quickly find that it cannot operate two kinds of portfolio management. Using a lean portfolio, you support agile teams without harming waterfall projects that you still have. But old-style, high-churn portfolio management will seriously clash with agile teams.

The four pillars thinking tool can help managers ask the right questions early on as they monitor agile teams.

Lessons for Organizations and Teams

Common Mistakes

Several of the unfortunate patterns described here are very common in failed agile endeavors. The most pervasive ones are discussed briefly in this section.

Allowing Issues to Compound

The Comet team got into quite a difficult situation even though it was in good shape when the external coaching ended. Once problems accumulate to this degree, it is very difficult to sort them out — too much so for an organization still getting used to agile. The four pillars thinking tool can help managers ask the right questions early on as they monitor agile teams.

Agile Adoption Sponsorship at Too-Low Level

PHI ran an agile adoption program with sponsorship ping-ponging back and forth between IT and product delivery. This is a pattern that is very common in failed agile adoption programs. Sponsorship has to exist at a level spanning the whole lean value stream — the set of activities that senses customer needs and delivers completed products to the customer. The level where agile sponsorship exists is where suboptimization will occur. For PHI, the highest sponsorship achieved was with the senior VP of product delivery, a sibling to IT. Therefore, suboptimization of the agile effort surfaced there, and it blocked Randy's efforts to get the agile adoption project back on track. At PHI, the lowest level of management spanning customer need to product delivery was the COO. Agile adoption sponsorship needed to be owned at that level.

Assuming Soft Skills Are Optional

Too often it is easy to dismiss agile management control points when they involve so-called soft skills like good facilitation or fringe-sounding ideas like team autonomy. In my years of agile coaching experience, it is this — the most neglected area — that produces the biggest returns for the business, provided the barriers between departments are addressed. I am convinced the ability to use team autonomy and group decision making in an optimal way will be what separates the top companies from the also-rans.

No Customer in the Loop

Too often I hear from newly agile teams that they "tailored" the methods because they couldn't get anyone from marketing or product management to agree to act as product owner. Cutting out the voice of your customer is a very bad way to customize agile practices. Yes, each company will implement the practices somewhat differently, and that is good. There is more than one way to achieve this, but it is vital that the team members can interact with someone who has the final word on what to build and who will be accountable for those decisions. Roles like Scrum product owner, lean style product champion, or product manager can fill the bill. Find a way to keep the customer's voice in the development loop.

Too-Fast Creation of New Agile Teams

It is understandable that businesses starved for high-quality IT outputs would want to open a floodgate of agile teams as soon as they see proof that the claims are true and work in their company. This tipping point occurs every time an agile pilot project succeeds. In a small shop with only a few teams, it causes some growing pains but no serious damage. In a large company that has not carefully planned its agile adoption as a project in its own right, it can be an outright train wreck as we saw with PHI.

New agile teams should be started only as the organization proves that it can adequately support the existing ones. Going step by step organically will allow you

to encounter new issues at a pace with which you can cope. A practice that I have found useful in iteration retrospectives is for the team to divide its list of open issues into two sets: the ones that the team can address by itself and the ones where the team needs outside support such as extra seat licenses or more time from its product owner. Then that second list is fuel for the agile adoption program to do its next increment of organizational changes.

Poor Use of Consultants

An agile adoption program has to sort out its philosophy early and then market it internally. PHI piloted agile without having actually sponsored that pilot. Then enthusiastic analysts and technical leads gravitated to a different methodology (EUP) and went into a lengthy study of it. Plenty was spent bringing in consultants during this period, only to have management lose faith in them and revert to the method that had actually been piloted. Then plenty was spent on Scrum consultants when they had been given no clear role in process leadership.

It's probably not useful to recommend that the COO should have simply commanded some methodology to be piloted and then rolled out. The COO is not likely to know about agile in the first place and would regard it as IT's territory. A more reasonable scenario would be this: After the Scrum pilot that Ben initiated on his own, the COO should inquire into the nature of how Scrum works. Then seeing that it spans the departments that report to him, he takes ownership of rolling it out further. This necessitates time to bring in the right experts, but then after the key decisions are made, a progressive series of pilots are run. Then an organically expanding set of agile team rollouts commences. There is no "big bang" rollout of teams.

Once Scrum consultants arrived, there should have been process goals set by PHI management (e.g., make the team's story backlog and stories completed per iteration visible; get agile testing frameworks into place; track the numbers of unit tests passing or failing with each build). The teams would need and want the coaches' help to achieve these goals. These process goals would be demonstrably supporting business goals. It should never be the case that teams feel they are getting dissonant messages from Scrum consultant coaches and their own managers. At PHI, the teams were hearing "do it right" from the coaches and "do anything but do it fast" from their managers. No wonder Randy saw the coaches' role reduced to "drive-by advice givers."

Making the Transition from Pilot to Enterprise Rollout

The way that pilot teams are run will vary according to the organization, but the guiding principle is that you do not want the initial trial to fail for reasons unconnected with the agile process. So it shouldn't fail because the right skills were missing from the team, or because it was a joint venture with a bad partner, and so on. Clearly you want the first pilot to have few other elements that can go awry. A second pilot might ask, "Can we run an agile project where part of our team is at another location?"

In this way, pilots should be a series of experiments the agile adoption leaders are mining for signs of possible clashes with other departments or with company policies or for other unforeseen issues.

Pilot success can mask other problems. Pilot projects are run in ideal conditions (handpicked staffers, available tools, prompt attention to issues) so that only the new methodology is under test. But these other supports tend to be forgotten when rolling the process out generally. You must account for this. Subsequent teams may need more training than the pilots needed. They may need organizational structures added to ensure prompt attention to issues, for instance, a wiki monitored by the agile adoption sponsor to record all blockages the teams experience.

As already mentioned, a pilot project doesn't "load test" the organizational structures, so care has to be taken to project the impact that multiple teams will have. The best way to mitigate your risks is to avoid having a mass rollout of new teams. Steady organic growth is the best way to manage the risks.

Walk Before You Run

Agile development is based on a "pay as you go" ethos. For example, a team should not agree to a quantity of work for the upcoming iteration until it has already accomplished the same quantity of work in a recent iteration.[16] If the team increases its speed through a new technique, then in some iteration it'll accomplish more than was promised. Only then can the team commit to doing that higher amount of work in a future iteration. This gives a conservative counterbalance to the tendency to overestimate what can be done. By ensuring that work commitments are backed up by proven ability to work at that speed, a team guards its credibility.

The agile adoption program needs some of this same conservative bent. A rule to ensure that new teams are started only when they can be sustained properly would be helpful. It might look like this:

Preconditions for starting a new agile team:

- No existing team has had blockages lasting more than a day, in the most recent iteration.
- All proposed team members can be dedicated 100% to the project without getting pulled off.
- A product owner is available and has already been trained in agile requirements work.
- Issues identified by existing teams will not be compounded by creating a new team.

If new teams are created one at a time, then unforeseen problems will surface either as blockages or as issues the existing teams call out and thus act to throttle the start of still more teams until they can be supported.

Agile Adoption Must Be Run as a Project

The best way for a project to fail is if the project has no one in charge and has no authority to do anything. Most agile adoption programs exist only as an idea held by one or a few people. All efforts are focused on getting this or that team proficient in agile skills, but there is no charter for doing work that supports agile teams generally. Such work includes tracking metrics across agile projects, providing agile awareness training to those affected by agile teams (but who are not members of them), and anticipating strain points between agile teams and other departments and taking steps to mitigate that.

The agile adoption project's mission could be stated this way: We will change the way we do project management, project portfolio management, software development, and test to be in line with lean principles. Furthermore, we will monitor the points where this conflicts with the organizational backdrop and will change the organization to best support the new teams.

CONCLUSION

It is ironic that PHI's new CEO called for predictability in reaction to his perception that agile software development is unpredictable. The organizational systems were reeling from the shock of launching dozens of agile teams at once without the underpinnings in place to support them. Would a properly run agile adoption program be predictable? It's impossible to predict and plan every step in changing a large company's business processes. But when we cannot predict, we have to be able to adapt. This need for adaptability is at the core of lean management ideas. A well-run agile adoption program would plan pilot projects, learn from their experience, and plan further agile projects if the pilots succeeded.

We can expect that virtually all the organization's structures will be affected by a shift to lean and agile practices. But what exactly should be changed and when? As agile teams are formed and begin operating, they will place varying levels of strain on organizational structures, and that's the signal to make a change to relieve that strain. An example is a team that achieved a significant project delivery by every member doing excellent work. Then the team's project manager is asked to rank the members so that the top person gets a bonus and the bottom one gets a performance warning. The forced ranking would undermine the team, so this should be questioned and changed. Better yet, as a planning exercise the agile adoption project should have anticipated this and many other places where agile values clash with company practices. Then they may make changes before problems arise.

All changes do not have to come incrementally from the needs of agile teams. Top-down guidance of the agile adoption project should make use of lean concepts like value stream mapping and learning to "see waste." These should be part of the agile adoption program because lean management practices support agile practices.

Agile is not just new software development techniques or new project management techniques. It is that and more; it is an overall management philosophy for an organization. Agile points the way to applying lean thinking to software-intensive parts of product development as well as business process improvement. PHI's managers had adopted agile practices but had not yet grasped its management philosophy.

The biggest gains of all come from management innovations. Management innovation is very hard to copy because it is almost invisible from the outside. Practices are easy to see and copy, but copying them seldom produces sustainable benefits within an incompatible system of management. Lean businesses are radically different from traditional businesses in almost every respect. This is why it is so difficult to move an organization fully to lean and agile methods. It's also why big rewards still await those who get there first.

ENDNOTES

[1]The term "agile" is used to mean any of the agile methodologies or frameworks such as Scrum, XP, DSDM, and Crystal. They are characterized by short iterations (two to four weeks), thorough (automated) testing prior to delivery, participation of customers in the development feedback loop, and teams empowered to improve their process continuously.

[2]The lean pull system is described nicely in the article "Lean Programming" by Mary Poppendieck (www.poppendieck.com/lean.htm). Refer to the section entitled "Lean Rule #4: Pull From Demand (Decide as Late as Possible)."

[3]The term "stakeholders" is intended to include customers, or those who represent them, representatives of other groups within the company that are affected by the agile project, and groups whose cooperation is needed by the agile team.

[4]Van Schooenderwoert, Nancy. "Embedded Agile Project by the Numbers with Newbies." Paper presented at *Agile 2006*, July 2006, Minneapolis, Minnesota, USA (www.leanAgilepartners.com/publications.html). The author found many other agile teams who had similar successes but had not produced detailed metrics. They typically said that they don't track bugs because they're so rare: maybe one in a month. They sometimes could not recall when they last had a bug.

[5]There are many factors involved beyond just the number of tests. The quality of tests matters, as well as how easy or difficult it is to change them. The key point is that there is no natural, obvious stopping point when creating tests. They can be as fine- or coarse-grained as desired.

[6]Independent testing external to the team is a separate matter and is not being ruled out by these statements. In a lean organization, each workgroup should routinely produce defect-free outputs. Ideally, final inspection or quality control should only find defects that could not have been prevented by the team. One of W. Edwards Deming's management principles is "Cease dependence on inspection to achieve quality. Eliminate the need for inspection on a mass basis by building quality into the product in the first place."

[7]A quote from Gary Hamel's "The Why, What, and How of Management Innovation" (*Harvard Business Review*, February 2006) is appropriate here: "Only after American carmakers had exhausted every other explanation for Toyota's success — an undervalued yen, a docile workforce, Japanese culture, superior automation — were they finally able to admit that Toyota's real advantage was its ability to harness the intellect of 'ordinary' employees."

[8]For more on W. Edwards Deming's 14 key points for management, see the entry on Wikipedia (http://en.wikipedia.org/wiki/W._Edwards_Deming).

[9]There are techniques to achieve higher velocity, such as using multiple agile teams for the project, changing composition of the team, and so on. These are beyond the scope of this report.

[10]Names have been changed, but this is a true story.

[11]An agile "story" can be thought of as a feature, but the size of stories has to be limited. This can sometimes prompt teams to create an infrastructure story — something well defined and necessary but that the customer does not see. An example of an infrastructure story is rearranging a database to allow the addition of new tables.

[12]Value stream mapping is a key tool in lean thinking to view the steps in creating a product or service. Introductory material can be found on Wikipedia (http://en.wikipedia.org/wiki/Value_Stream_Mapping).

[13]The engineering skills referred to here can be summarized as the practices of XP. These include continuous integration, fully automated builds, automated unit test suites, and test-driven development. Scrum's inventors acknowledge that Scrum is a framework that does not dictate the engineering methods, but that Scrum should be combined with practices designed to create defect-free software.

[14]The term "bug-free software" is intended here to mean that the software is routinely, but not always, found to contain no defects after it leaves the development team. Human error will always exist, but if teams have easy-to-run tests covering all the expected defects, then new ones will be rare. New ones will also point the way to improved tests. The result is that virtually no time is wasted doing rework. Lean work cells have demonstrated this idea for decades in manufacturing companies that use lean thinking. The ability of teams — or work cells — to routinely produce defect-free work is the bedrock that the rest of the lean system rests upon.

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

[16]The assumption is that iterations are all the same length. That isn't always the case, but if not then we can speak of a given amount of work per week.

ABOUT THE AUTHOR

Nancy Van Schooenderwoert is a Principal Coach at Lean-Agile Partners, Inc. Lean-Agile Partners demonstrates to their customers ways to use and adopt agile practices in their organizations. They "teach their customers how to fish" by training customer staff to coach agile teams. Working with both their customers' management and technical teams, Lean-Agile Partners helps companies design, plan, and implement their agile adoption programs. Customers get the benefit of Lean-Agile Partners' experience while staying in control of their own agile transformation.

Ms. Van Schooenderwoert is an agile enterprise coach with over a decade of experience applying agile practices as an engineer, manager, and consultant. She shows executives how to apply agile and lean principles to all their business processes, not just software development. She has led agile change initiatives in life-critical, highly regulated industries and coached clients in the art of agile technical and management leadership. Her experience spans embedded software development and applications in aerospace, factory automation, medical devices, defense systems, and financial services. Ms. Van Schooenderwoert can be reached at NancyV@LeanAgilePartners.com or via www.leanagilepartners.com.

About the Practice

Agile Product & Project Management Practice

Cutter Consortium's Agile Product & Project Management practice provides you with information and guidance — from experienced, hands-on Senior Consultants — to help you transition (or make the decision to transition) to agile methods. Led by Practice Director Jim Highsmith, Cutter's team of experts focuses on agile principles and traits — delivering customer value, embracing change, reflection, adaptation, etc. — to help you shorten your product development schedules and increase the quality of your resultant products. Cutting-edge ideas on collaboration, governance, and measurement/metrics are united with agile practices, such as iterative development, test-first design, project chartering, team collocation, onsite customers, sustainable work schedules, and others, to help your organization innovate and ultimately deliver high return on investment.

Through the subscription-based publications and the consulting, mentoring, and training the Agile Product & Project Management Practice offers, clients get insight into Agile methodologies, including Adaptive Software Development, Extreme Programming, Dynamic Systems Development Method, and Lean Development; the peopleware issues of managing high-profile projects; advice on how to elicit adequate requirements and managing changing requirements; productivity benchmarking; the conflict that inevitably arises within high-visibility initiatives; issues associated with globally disbursed software teams; and more.

Products and Services Available from the Agile Product & Project Management Practice

- The Agile Product & Project Management Advisory Service
- Consulting
- Inhouse Workshops
- Mentoring
- Research Reports

Other Cutter Consortium Practices

Cutter Consortium aligns its products and services into the nine practice areas below. Each of these practices includes a subscription-based periodical service, plus consulting and training services.

- Agile Product & Project Management
- Business Intelligence
- Business-IT Strategies
- Business Technology Trends & Impacts
- Enterprise Architecture
- Innovation & Enterprise Agility
- IT Management
- Measurement & Benchmarking Strategies
- Enterprise Risk Management & Governance
- Social Networking
- Sourcing & Vendor Relationships

Senior Consultant Team

The Cutter Consortium Agile Product & Project Management Senior Consultant Team includes many of the trailblazers in the project management/peopleware field, from those who've written the textbooks that continue to crystallize the issues of hiring, retaining, and motivating software professionals, to those who've developed today's hottest Agile methodologies. You'll get sound advice and cutting-edge tips, as well as case studies and data analysis from best-in-class experts. This brain trust includes:

- Jim Highsmith, Director
- Sanjiv Augustine
- Christopher M. Avery
- Paul G. Bassett
- Sam Bayer
- Kent Beck
- E.M. Bennatan
- Tom Bragg
- David R. Caruso
- Robert N. Charette
- Alistair Cockburn
- Jens Coldewey
- David Coleman
- Ken Collier
- Ward Cunningham
- Rachel Davies
- Doug DeCarlo
- Tom DeMarco
- Lance Dublin
- Khaled El Emam
- Kerry F. Gentry
- Chet Hendrickson
- Sid Henkin
- David Hussman
- Ron Jeffries
- Joshua Kerievsky
- Bartosz Kiepuszewski
- Brian Lawrence
- Tim Lister
- Michael C. Mah
- Lynne Nix
- Siobhán O'Mahony
- Ken Orr
- Patricia Patrick
- Roger Pressman
- James Robertson
- Suzanne Robertson
- Alexandre Rodrigues
- David Rooney
- Johanna Rothman
- David Spann
- Rob Thomsett
- John Tibbetts
- Jim Watson
- Robert K. Wysocki
- Richard Zultner