Cutter IT Journal

The Journal of Information Technology Management

The Viral Growth of Kanban in the Enterprise

Replacing a Bad Process Produces Better Results

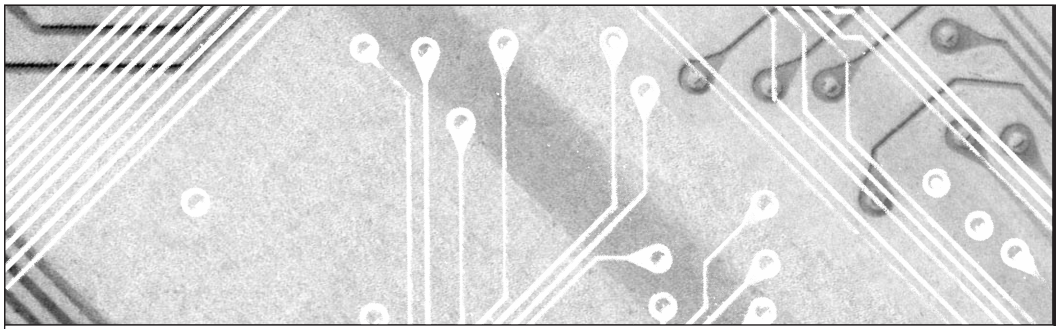
An organization that is operating poorly is better off getting rid of its current process and taking on an entirely new one. That way the bad habits are eliminated at once, and things are done better from day one.

Evolving a Bad Process Produces Better Results

Taking your current process as a starting point and gradually improving it significantly reduces resistance to change. It also reduces risk because a small bad change is easier to fix and a small good change is easier to achieve.

REPRINT

CUTTER CONSORTIUM



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, Christine Davis, Tom DeMarco, Lynne Ellyn, Israel Gat, 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.

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
E-Mail (Be sure to include for weekly *Cutter IT E-Mail Advisor*)		

Fax to +1 781 648 8707, call +1 781 648 8700, or send e-mail 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 Masa K. Maeda

MOTIVATION

Let's take two premises as a starting point: time is relative, and technological advances have effectively made many things happen faster. An expert is not necessarily someone with lots of gray hair or very little hair, and a proven technology is not necessarily one that has been around for a long time.

The Kanban method (not to be confused with the kanban developed as part of the Toyota Production System) was originated in 2004 during an attempt to rescue a small IT team at Microsoft that was operating so poorly — despite being certified as CMMI Level 5 — that it was about to be shut down. While assisting the team, David J. Anderson developed a method whose impact was so impressive that the team won Microsoft's Engineering Excellence Achievement Award just five fiscal quarters later.[1] Anderson applied and further matured the method at Corbis in 2006. When he showed his work to some well-respected people in the lean and agile communities, they concluded it was a new and powerful method. Anderson published his work,[2] and in 2008 Kanban's actual growth began.

A method that made its public debut just three years ago could be considered by many as very high risk and unproven. But if we consider that ISO took a couple of decades to be developed, published, and used worldwide, while the CMM took around 13 years to achieve similar adoption and agile just six, then the pattern suggests that good things can happen more quickly now, as technology allows us to communicate and collaborate more effectively. In the September 2010 issue of *Cutter Benchmark Review* on Kanban,[3] I led a worldwide survey whose results indicate that Kanban is already being adopted in all regions of the world and by companies from very small startups (fewer than a half-dozen employees) to very large businesses (more than 50,000 employees). Furthermore, around 60% of reports indicate an increase in productivity, around 65% report an increase in quality, and around 70% report an increase in customer satisfaction from adopting Kanban as a way of working.

CHALLENGES

I think Kanban is facing, and will face for a while, some unique challenges until it becomes better known:

- **What's in a name?** The first challenge is the name itself. As I bring Kanban offerings to potential customers and am about to begin talking, time and time again I get that "Why is he telling us this is a new method when kanban has been around for decades?" look, because naturally they are thinking of kanban in manufacturing. The Kanban method is not the same as kanban for manufacturing, although they share the teachings of W. Edwards Deming as a common base. The second challenge has to do with its origin. The Kanban method is a unique development influenced by the work of Deming (as noted), Eli Goldratt, Donald D. Reinertsen, Mary and Tom Poppendieck, the Agile Manifesto, the Declaration of Interdependence, and some kanban from manufacturing. Such a pedigree motivates some people to call Kanban an agile methodology, a second-generation agile methodology, a lean methodology, or a lean-agile methodology (my preferred term). Such diverse classification tends to confuse those who are making their first contact with Kanban.
- **Counterintuitiveness.** Kanban is counterintuitive in a number of respects. For example, Kanban limits the amount of work in progress, which actually allows you to get more work done and with better quality. This makes most leaders stop for a moment and wonder whether such a thing is actually possible. As a result, some leaders decide not to try Kanban, whereas others see the potential and try it out.
- **Changes.** Kanban is about gradual change — a gentle but highly effective approach. We are so used to big changes and radical changes … the modern world is fast-paced, after all. So a gradual, incremental change could be a challenge to the patience of some. Patience will pay off, though, as results show an actual acceleration in the improvements the organization needs.

- **Unusual.** We are used to replacing methodologies or processes to make improvements. The Kanban method is neither a project management methodology nor a process. It brings change and improvement to the way some activities — such as communication, coordination, planning, analysis, and reporting — take place. In other words, it will help you incrementally improve your current process, *not* replace it. Adopting the Kanban method as a plug-in to an existing process instead of as a replacement for it may strike some as a bit odd.

Now, where does the rubber meet the road? Is Kanban really delivering? Are adoption results astounding, modest, or poor? Is Kanban for your organization? This issue of *Cutter IT Journal* brings you Kanban experiences from diverse parts of the world to help answer those and other questions you may have.

KANBAN IN A NUTSHELL

Kanban was developed in response to the need to reduce resistance to change, handle risk and variability effectively, exercise continuous improvement, and improve the quality of work life. Improvements focus on:

- Visualizing and managing the workflow
- Limiting the amount of work in progress (WIP)
- Making the process policies explicit (e.g., in the form of classes of service to define the behavior of, say, tasks, defects, and urgent tasks)
- Identifying opportunities for improvement and collaborating to address them

The Kanban wall shown in Figure 1 is a visualization of a process that flows from left to right. It indicates the maximum amount of work in progress at each stage, differentiating classes of service by color (and with a unique lane in the case of expedited tasks). Notice that tasks are not clustered by blocks all together at one same stage but rather flow through the process as each individual task is finished at each stage. Policies such as classes of service are put into practice upon mutual agreement among all the stakeholders involved. A Kanban wall does not have a fixed shape. It reflects the unique process for each project and changes as the project itself matures.

IN THIS ISSUE

In our first article, David J. Anderson, the creator of the Kanban method, and Arne Roock, a lean and Kanban coach in Germany, take Kanban's adoption in that country as a starting point to discuss aspects of the method that influence its adoption around the globe. The authors discuss how Kanban went from being virtually unknown to "one of the most discussed topics in software development" in Germany and list what they consider to be the five main reasons responsible for its introduction there. I would dare to say that diverse combinations of those reasons are also applicable to most other countries in which Kanban is being adopted. The authors warn us that Kanban's growing popularity can lead to what they call a "Kanban veneer" — implementations in which the visible elements of Kanban (the Kanban board, WIP limits, stand-up meetings, etc.) are embraced but no true pull system develops. Such implementations, they say, leave Kanban's full potential untapped. The last section of the article offers some eye-opening data that shows

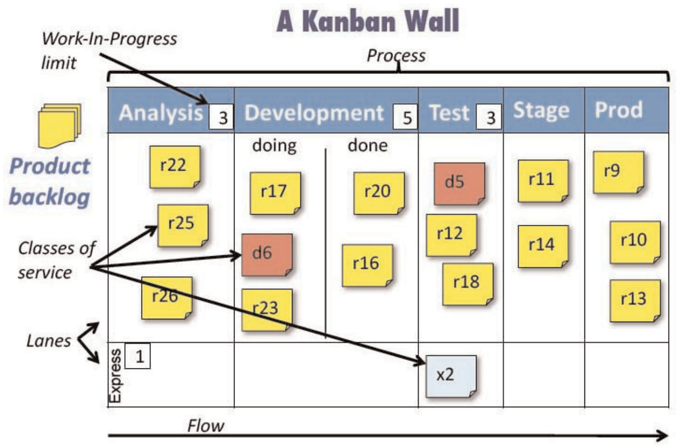


Figure 1 — A Kanban wall.

how corporate and national cultures influence the speed of adoption.

Kanban's adoption worldwide has been so quick that there is a risk some adopters will misinterpret it. Alan Shalloway, founder of Seattle-based NetObjectives, begins his article with a comparison between Kanban and agile methodologies, such as Scrum and XP, distinguishing Kanban from the others by calling it a "second-generation" agile method. His paper has a strong enterprise flavor — unlike earlier "team-based" methods, Shalloway says, Kanban addresses the entire value stream. He discusses the four faces of Kanban, describing its use as a lean-agile team method, a transition management system, a means of learning, and an aid to product portfolio management. He also addresses various misconceptions that have sprung up about Kanban. These discussions are helpful in understanding what Kanban is and is not. And if an organization is still unsure, Shalloway closes with a test for determining whether or not an organization is actually doing Kanban.

Next, Dan Verweij, a program manager at ASR Insurance in the Netherlands, and Olav Maassen, a principal consultant for Xebia (also in the Netherlands), tell us about the adoption and evolution of Kanban at ASR. This case is a very good example of viral growth in the enterprise. They begin by relating how the company had adopted Scrum for project work but discovered that the method wasn't a good fit for its maintenance and operations (M&O) teams. So in August 2009, ASR launched a pilot project to introduce Kanban as the M&O teams' new way of working. Eighteen months later, 160 people in 18 teams are successfully using Kanban at ASR. The authors cover the benefits of Kanban and the cultural, business, and communication difficulties ASR confronted during adoption. They conclude with a set of their experience-gained recommendations for adopting Kanban.

In our next article, Roland Cuellar, director of program management at Three Pillar Global in Washington, DC, gives us a case study on Kanban implementation at a help desk organization. He discusses the unique challenges of a help desk organization's work (what he calls the "unplannable") and explains why traditional and even agile methodologies don't quite fit in that kind of organization. He then details how the team he was working with was able to obtain immediate benefits simply by implementing some of the practices of Kanban and how those benefits increased over time. He closes with a discussion of the challenges the team confronted and the evolution of their implementation.

Last but not least, Siddharta Govindaraj, founder of Silver Stripe Software in Chennai, India, and Sreekanth Tadipatri, an agile coach based in Bangalore, India, present their experiences using Kanban in outsourced projects. They begin by discussing the limitations of typical agile methods (e.g., Scrum) in such environments and offer guidelines, punctuated by experience reports, for applying Kanban to outsourced projects. The authors identify pitfalls to avoid and address some cultural challenges that could derail Kanban adoption. This discussion is a good complement to the cultural aspects touched on by Anderson and Roock.

NEXT STEPS

We hope this issue offers you diverse perspectives and enough information to help you make a decision on your next steps regarding Kanban. It is also our hope that you feel encouraged to further explore this recent but rapidly growing and highly effective method.

ENDNOTES

[1]Anderson, David J. *Kanban: Successful Evolutionary Change for Your Technology Company*. Blue Hole Press, 2010.

[2]Anderson. See 1.

[3]Piccoli, Gabriele (ed.). "Kanban for Project Management: Should We Buy In?" *Cutter Benchmark Review*, Vol. 10, No. 9, 2010.

Masa K. Maeda is a Senior Consultant with Cutter's Agile Product & Project Management Practice whose primary focus is on Kanban and lean-agile project management. Dr. Maeda is also the founder of Shojiki Solutions, a coaching and training firm based in the San Francisco Bay Area that focuses on Kanban, lean-agile, and value innovation, and an associate with David J. Anderson & Associates. His over 24 years of experience include work at Apple Inc. doing development for the Mac's operating system and as a founding team member at startups in the life sciences (Ingenuity Systems), online video (Vuze Inc.), and online socialization (When.com) industries. Dr. Maeda was also a consultant who developed AI prototypes for commercial applications and core technology algorithms for Justsystems Corporation, a leading software company in Japan. In Mexico, he worked for the financial and services industries.

Dr. Maeda founded the Bay Area chapter of the Limited WIP Society and the Mexico chapter of the Agile Project Leadership Network. He will begin teaching value innovation at UC Berkeley Extension this year. He has a PhD in software engineering and artificial intelligence and a master's degree in intelligent systems engineering and information science from the University of Tokushima in Japan and a bachelor's degree, with honors, in computer engineering from the National University of Mexico. Dr. Maeda can be reached at mmaeda@cutter.com.

An Agile Evolution: Why Kanban Is Catching On in Germany and Around the World

by David J. Anderson and Arne Roock

In April 2009, we conducted the first public Kanban training seminar in Germany — with two paying participants! At the time, Kanban was largely unknown in Germany; we assume that no more than a handful of people had ever heard of it. No German Wikipedia entry existed for Kanban, let alone German-language blogs or professional articles on the subject.

Today, a mere two years later, the situation has changed completely. Without doubt, Kanban is one of the most discussed topics in software development, and not just in the agile community. At each of the larger software conferences, there will be at least one talk focusing on Kanban, and IT magazines regularly publish articles and experience reports about it. We have encountered a growing number of people who already implement Kanban in their teams.

How can one explain this rapidly increasing dissemination? For what reasons do more and more teams and organizations rely on Kanban? What pitfalls can prevent a successful introduction of Kanban? And do the phenomena we observe in Germany apply only to this country? To what degree do they differ from the popularity and implementation practice of Kanban in other European countries and on other continents?

KANBAN: THE STATUS QUO

Contrary to popular belief, Kanban is neither a development process nor a management framework. Rather, Kanban is a method for advancing incremental, evolutionary changes in IT organizations through the adoption of kanban[1] systems, which limit work in progress (WIP) and create a pull system where new work can only be started as existing work is completed. The Kanban method is based on three foundational principles:[2]

1. Start with what you do now.
2. Agree to pursue incremental, evolutionary change.
3. Respect the current process, roles, responsibilities, and titles.

In addition to these principles, five core practices have been observed in organizations that have been successful in achieving a culture of continuous improvement using Kanban:

1. Visualize the workflow.
2. Limit WIP.
3. Manage flow.
4. Make process policies explicit.
5. Improve collaboratively (using models and the scientific method).

Kanban suggests that we start out with the actual state of our process — no matter what this may look like — and proceed to further develop it in small steps. We cannot foresee where our path will take us, and there is no predefined final state. If Kanban is understood correctly, it will result in an organization that experiences continual improvement as time goes by and as the organization becomes more familiar with its techniques. In such a kaizen (i.e., continuous improvement) culture, problems are individually identified and solved. The team will consistently make suggestions for improving the process and test these suggestions.

As evolution in nature creates highly specialized species (e.g., the bird-of-paradise, with its elaborate and long tail feathers, or the long-necked giraffe), Kanban leads to the creation of highly specialized systems that are optimized to function in their respective contexts. Copying and applying a working Kanban system to another context will very likely not yield the desired result.

A SHORT HISTORY OF KANBAN'S DISSEMINATION IN GERMANY

In less than two years, Kanban has emerged almost from thin air to become a method that is used by a growing number of diverse teams and organizations. Besides Kanban's various advantages and the publication of David's book *Kanban: Successful Evolutionary*

Change for Your Technology Business[3] in April 2010, the rapid proliferation of this method can be explained by three phenomena:

1. **Conference reports.** Some early adopters began to use Kanban in spring 2009 and reported on the method at several conferences. The key impetus for these early adoptions was Kanban's focus on flow and the absence of obligatory iterations. For example, Markus Andrezak, who pioneered Kanban at mobile.de, always felt that the WIP of a Scrum sprint was still too much to manage in meetings with the company's offshore partner.
2. **Influential articles.** A number of IT magazines developed an early interest in Kanban, which led to the publication of various articles about this method. First and foremost, the popular IT magazine *OBJEKTspektrum* published a special issue about lean and Kanban in March 2010, as it recognized the tremendous potential of Kanban, which was then relatively unknown. Besides introductory articles, the same magazine also featured two experience reports: one by mobile.de (the biggest German Internet sales portal for used cars) and one by XING (a professional and social networking Web site similar to LinkedIn). Kanban's popularity was further fostered by a comprehensive feature published in the June 2010 edition of the monthly computer magazine *iX*, as well as by an online article on Heise Developer Channel. The latter is one of the most frequently called-up articles ever posted by the channel.
3. **The insufficiencies of Scrum.** Kanban became popular in Germany at a time when many companies had been using Scrum for several months and found it — for various reasons — to be very challenging. The evolutionary character of Kanban was and is attractive to these organizations, which face the task of improving (partially incomplete) Scrum implementations in small increments. While Scrum blazed a trail for agile in Germany and provoked a lot of interest in agile ideas, it appears that it is not the best cultural fit for all companies. It might be said that the adoption of Kanban is partly a learning effect of up to 10 years of experience with Scrum (and XP).

The quote "It is difficult to make predictions, especially about the future" (attributed to everyone from Niels Bohr to Yogi Berra) definitely applies to Kanban. However, we feel it is safe to assume that Kanban's popularity and usage will continue to grow in Germany in 2011. David's *Kanban* book is now available in German, and it's supplemented by an additional chapter containing an experience report about portfolio management with Kanban at mobile.de. The publisher reported that interest in the first week of sales was greater than for any other technical book the company had previously published. This is a further indicator that interest in Kanban continues to grow strongly in Germany. Moreover, Siemens and at least one other leading international German corporation are currently introducing Kanban. This should further help to establish Kanban as a suitable method for big businesses. In addition, several large enterprises, among them a major German insurance company, have explicitly added Kanban know-how to their requirements for hiring internal coaches.

The key impetus for the early adoptions was Kanban's focus on flow and the absence of obligatory iterations.

THE MAIN REASONS FOR USING KANBAN

There are five main reasons for the introduction of Kanban in Germany:

1. Iteration Problems

Like agile methods, Kanban firmly relies on simplicity, regular feedback, direct communication, empowerment of employees, and teamwork, but it does not require the use of the timeboxed increments referred to as "iterations" in most agile literature. Due to Scrum's overwhelming success in the last few years, many organizations had high expectations regarding this method before they realized that sprints (the Scrum term for a timeboxed increment of functionality) are not well suited to their purposes. This has proven particularly true in the media industry, which globally has been a significant early adopter of Kanban, with examples such as the BBC, Sky, IPC Media, the *Financial Times*, and Lonely Planet.

When organizations attempt to adopt agile methods, a mismatch between the process prescription and the nature of their domain can cause them to reject the new approach and revert to their previous (typically "conventional" or "traditional") way of doing things. This has been particularly true with Scrum, where the community has encouraged adoption "by the book" and publicly discouraged adaptation. Kanban, on the other hand, offers the opportunity to build on what is already working well and design out existing problems and challenges one at a time.

Teams that frequently run into considerable problems with timeboxed iterations include those performing ongoing maintenance and production support or shared organization resources such as architecture, security, user experience design, database administration, business intelligence, and devops (or configuration management). Such teams have to tackle vastly different tasks (with regard to scope, origin, technology, urgency, etc.), and their work is dominated not only by unpredictable events, but often by emergencies as well. Usually even one-week sprints are too restrictive for them. At the same time, it is becoming clear that some Web startups feel hampered by iterations, too, because they perceive a flow-based work method to be more favorable (and more innovation-friendly) than strict iterations. This is exemplified by the strong adoption of Kanban among such online service firms as CityGrid Media, Constant Contact, Ultimate Software, XING, mobile.de, Spotify, BWin, and ICA.

Kanban's real potential will remain untapped until a real pull system is established and employees are empowered to make independent team-based decisions.

2. Evolutionary Changes

Large organizations in particular have a hard time making disruptive changes that would result in completely changed work methods and roles. In many cases, such changes will be enforced only if a company stands with its back against the wall — perhaps because an important project is about to fail or competitors are gaining. Cutter Senior Consultant Kent Beck shared this assessment during a lecture in Hamburg in October 2010: "The pace of change is what people are willing to accept. Only when an organization is close to death will it change very fast." Therefore, changes made in small increments — as with Kanban — are generally preferable.

3. Affinity with Lean

Top management — especially in big corporate groups — is often familiar with at least the basic ideas behind lean approaches and already know some lean concepts from production. Since Kanban is recognized as part of the lean body of knowledge, its chances of being accepted and supported by top management teams are good.

4. Portfolio Management

Kanban can be applied with good results for visualizing entire projects on a strategic level, speeding up their implementation, and improving the process for portfolio management. The latter explains why more and more organizations use Kanban to manage their portfolios, whereas single projects are run using Scrum. The fact that portfolio management with Kanban is comparatively popular in Germany can probably be attributed to Andrezak, who did some groundbreaking work in this field at mobile.de and early on made his experiences available to the public.[4]

5. Multitasking

During the past few months, it has become increasingly clear that Kanban is of interest to a large field of users who, for the most part, never dealt with lean approaches or agile methods before. We are referring to small and medium-sized Web agencies, which traditionally struggle with an extremely high degree of multitasking (often with individuals working on 20 or more tasks simultaneously). Kanban enables such firms to visualize the amount of work and they have to focus on a smaller set of urgent and critical tasks. This enables them to get more work done, faster, and to deliver it when most appropriate.

BEWARE "KANBAN VENEER"

The most common problem we observe with Kanban implementations can be described as "Kanban veneer." The workflow is visualized on a whiteboard, tickets are moved across the board, and daily stand-up meetings often take place, too. In this manner, a high level of transparency about the status quo of tasks and current problems can be obtained in very little time. Yet Kanban's real potential will remain untapped until a real pull system is established and employees are empowered to make independent team-based decisions about how to distribute work, how to solve problems, and how to improve processes.

In many cases, company management is either unable or unwilling to give up established command-and-control micromanagement and trust employees to make decisions. Managers are fearful that delegating decision making obviates their position. They fail to switch into a mode where they act as the system designer and focus on decisions about process policies rather than decisions about work in progress. At the same time, teams often fail to collaborate on kanban system design and policy setting. Instead, they take the current state for granted or leave all changes to the system designer.

Symptoms of a Kanban-in-name-only implementation include:

- A fast lane that is always full (meaning we're increasing lead time and losing predictability concerning the non-expedite tickets)
- A static system that does not improve (the board looks exactly the same today as it looked one year ago)
- Exceeded or constantly raised WIP limits (because the actual system capacity does not meet the demand and, instead, new tasks are repeatedly pushed into the system)

The term "veneer" is apt because only two core properties — visualization and WIP limits — are observed, and though visible from the outside, they are being undermined from within. Hence, a Kanban veneer exists, but a true pull system does not. Such a veneer prevents the formation of a reliable and predictable kanban system; it also suppresses discussion of change opportunities and therefore the development of a kaizen culture.

KANBAN ADOPTION AROUND THE WORLD

In the last two years, Kanban has been catching on around the world. In South America, examples have been reported in Argentina, Brazil, Chile, and Peru. The best known of these include Phidelis, a software company from Vitória, Brazil; CESAR, an outsourcing firm in Recife, Brazil, rated at CMMI Level 3; Huddle, an outsourcing firm in Buenos Aires, Argentina, that specializes in the media industry; and the IT department of Petrobras, the Brazilian state-owned oil company and one of the largest firms in South America. David's book is now available in Spanish and will soon be published in Portuguese.

The reasons for Kanban adoption in South America vary. Outsourcing firms appreciate the service-oriented nature of Kanban and desire a culture of continual improvement in order to achieve a competitive edge. Firms such as Phidelis like the flexibility to respond rapidly to changing customer demands as well as the cultural aspects of Kanban, which have enabled them to stay very small while capturing a very large market share. Larger firms seem to value the governance aspects of Kanban, which allow a better connection between shop-floor processes, market demands, and executive decisions. Firms such as Petrobras have been able to merge together multiple smaller teams into larger teams with greater labor flexibility and critical mass to respond to customer demand. This trend has been seen elsewhere at firms such as Constant Contact and Ultimate Software in the US.

Kanban is also increasingly adopted in the Benelux, with firms as diverse as several-hundred-year-old insurance companies in the Netherlands, government departments in Belgium, and smaller consulting and training firms using it for marketing and business development.

Kanban has perhaps its strongest market in Scandinavia. Sweden was the earliest adopter, with firms from large industrial groups, such as Volvo and Scania, to small mobile games and online gambling firms taking an interest. One of the best-known adopters is Spotify, the music streaming service. Examples of Kanban adoption also exist in Finland, Norway, Iceland, and Denmark, including one early case study reported at the Danish Ministry of Defense.

In the last two years, Kanban has been catching on around the world.

In South Africa, a large bank in Johannesburg reportedly adopted Kanban in 2010. And in the southern Pacific, there are reports of Kanban usage at Lonely Planet, the Australian travel guide publisher; at games development firms in Australia; and at the Ministry of Social Development in New Zealand.

During 2010 Kanban was also rapidly adopted in Israel. Reported case studies include Amdocs, Israel's largest indigenous software firm, famous for its telecommunications billing software; Internet firm answers.com; and software tools startup Typemock. What is remarkable about these examples is their breadth. From a small, less-than-20-person software tools vendor, to a 100-person Internet service, to a large software solutions vendor with tens of thousands of employees, Kanban is proving useful in diverse ways.

CULTURE AND KANBAN ADOPTION

Kanban has become firmly established on five continents in the last two years. This adoption has largely happened faster than the adoption of agile methods 10 years earlier. The early adopters of agile methods — the Swedes, the London-based banking industry, and software firms on the East and West Coasts of the US — were once again the earliest adopters of Kanban. Kanban offered them solutions to problem spaces where agile methods had proven a challenging fit.

What is more remarkable is the adoption of Kanban in other parts of the world and how rapid it has been. Kanban adoption in the Benelux is almost as prevalent as in Scandinavia. The same is true regarding demand for Kanban classes and firms offering such classes. This is remarkable, because the adoption of agile methods in the Benelux trailed that of Scandinavia by about six years.

Firms that had ignored the agile revolution have adopted Kanban.

Likewise, agile methods were slow to be adopted in the more conservative parts of Europe, such as Germany, France, Italy, and Spain, but again we see significant adoption of Kanban in the last two years in these countries, including by an automotive firm in Italy.

It is worth asking why this might be. Why is Kanban being adopted faster across a larger number of countries all around the world? It may be true that agile methods have blazed a trail and created a community and a market that have embraced the Kanban meme. However, it seems that Kanban appeals to a much broader market.

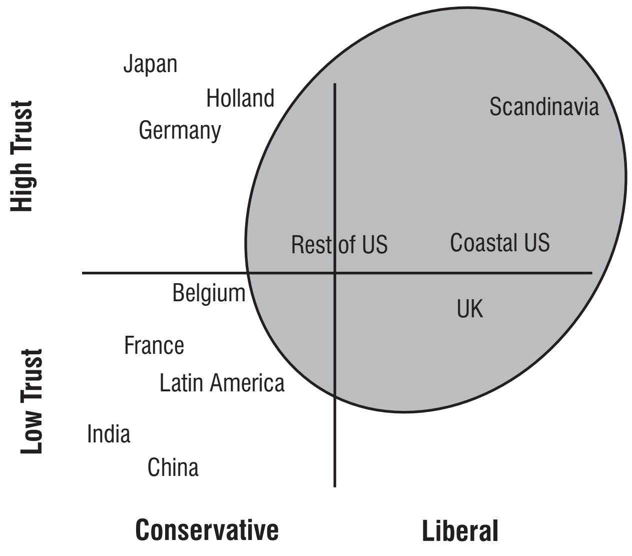


Figure 1 — Early agile adoption.

Firms with CMMI Level 3 and Level 5 appraisals have been adopting Kanban. Firms that had ignored the agile revolution, such as answers.com, have adopted Kanban. Kanban is finding a niche in devops and IT operations departments. It is also growing in popularity in industries with large numbers of specialist functions, such as games and investment banking, and with firms in rapidly changing domains such as media.

It appears that Kanban, which by its nature is designed to be adopted in a minimally invasive way that embraces the status quo and provides a framework to prompt slow but steady evolutionary change, is more appealing than agile to a broader spectrum of people and businesses. Kanban is popular with liberal-thinking innovators who work in high-trust cultures, but it appears to appeal equally to conservative, slow-moving industries and to professionals who work in bureaucratic, lower-trust organizations.

Figure 1 shows a two-dimensional space divided on two axes. The x axis spreads from ultraconservative (meaning very reluctant to embrace change) through to ultraliberal (meaning very enthusiastic to embrace new ideas). The y axis spans from very low-trust, low–social capital cultures through to high-trust, high–social capital cultures. The positions of various countries on the y axis are extrapolated from Francis Fukuyama's work in sociology.[5] The positions along the x axis are based more on our empirical observations from traveling the world and experiencing these cultures. So Sweden is a relatively high-trust country with a very liberal attitude toward business, while the Netherlands and Japan are very high-trust countries with a more conservative attitude toward business.

Figure 1 maps agile method adoption until about 2003. It is worth observing that agile methods were adopted in highly liberal cultures first. While agile methods need a high-trust culture in order to work, the existence of a high-trust culture did not correlate with early adoption. Agile adoption in Japan, the highest-trust country of all, is still extremely low after 10 years.

Using the same axes, Figure 2 maps the adoption of Kanban since 2007. Note that the adoption is wider and that the more conservative cultures have adopted it fairly rapidly. We attribute this difference specifically to the evolutionary, incremental nature of Kanban. It is more appealing in conservative cultures than the revolutionary approach required by some agile methods.

We can see that Kanban is still relatively unknown in China and India and that adoption of the method in this part of the world is much slower than elsewhere. Agile methods have also struggled in these countries. At this time, we can offer no explanation for this phenomenon. However, David is planning a trip to this region in 2011 to raise awareness of Kanban.

SUMMING UP

Kanban has spread rapidly in Germany during the past two years. On the one hand, Kanban is attractive to businesses that have problems with the iteration concepts required by many agile methods. On the other hand, Kanban is interesting, particularly for large companies, because changes are made in small increments and because of its affinity with lean approaches. On a strategic level, portfolio management with Kanban is becoming more and more common. Finally, we have observed another recent development; namely, small and medium-sized Web agencies beginning to adopt Kanban to tackle the multitasking problems that are particularly prevalent in this sector.

It appears that Kanban is catching on rapidly in parts of the world that have already embraced agile methods. Yet Kanban also appears to be offering an alternative approach to organizations that were struggling with agile adoption or had decided that agile methods were not for them, even though they still desired the benefits of agility and improved economic performance.

It is also noteworthy that Kanban adoption is occurring in a wide set of circumstances, from IT operations, devops, software development, and project management through to portfolio management. The nature of Kanban as an evolutionary, incremental approach to change management seems to be resonating with people and giving them hope that they can find a way to improve their processes and their organization's performance. The Kanban community has encouraged experimentation, and some of these experiments appear to be bearing fruit. Reports such as those from mobile.de can only encourage others to try. Consequently, in early 2011 we have considerable hope for further, faster, and wider adoption of Kanban both in Germany and throughout the world.

ENDNOTES

[1]In this article, we use the term "Kanban" (with a capital K) to denote the evolutionary change method that uses a kanban (small "k") pull system, visualization, and other tools to catalyze the introduction of lean ideas into technology development and IT operations.

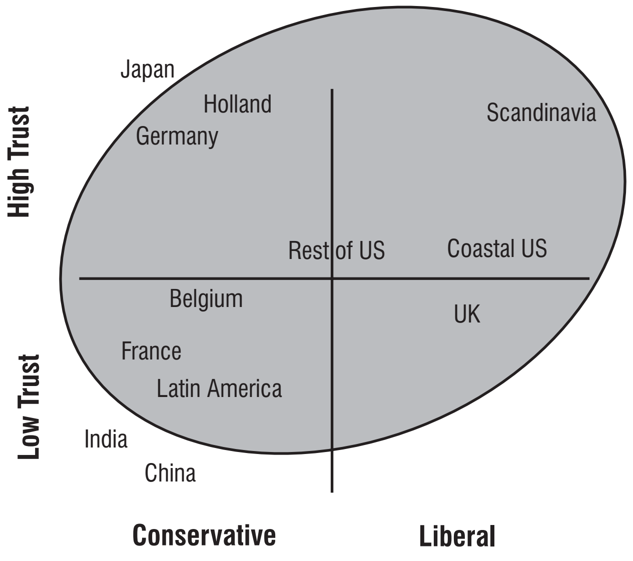


Figure 2 — Kanban adoption since 2007.

[2]Anderson, David J. "The Principles of the Kanban Method." David J. Anderson & Associates, 10 December 2010 (http://agilemanagement.net/index.php/site/the_principles_of_the_kanban_method).

[3]Anderson, David J. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

[4]Andrezak, Markus. "Feeding the Kanban — Portfoliomanagementbei mobile.international GmbH" (in German). In David J. Anderson, *Kanban: Evolutionäres Change Management für IT-Organisationen*, translated by Arne Roock and Henning Wolf. dpunkt.verlag, 2010.

[5]Fukuyama, Francis. *Trust: The Social Virtues and the Creation of Prosperity*. Free Press, 1995.

David J. Anderson has been a manager and leader of great software teams delivering cutting-edge software products since 1991. He has a successful track record of building progressively bigger teams capable of hyperproductive performance and superior quality. Mr. Anderson is a founder of the Lean Software & Systems Consortium, a nonprofit organization that seeks to bring a new level of professional conduct and capability to the software and systems engineering professions. He also helped create the Limited WIP Society, a loosely affiliated organization that encourages the development of a community of people using kanban (and other pull) systems in software engineering and project management. Mr. Anderson can be reached at dja@djandersonassociates.com.

Arne Roock works as a coach and trainer for it-agile GmbH in Germany. As an expert on lean and Kanban, he's been observing the evolution of Kanban in Germany over the past few years. Mr. Roock has written several papers on lean/Kanban and translated into German David J. Anderson's book Kanban: Successful Evolutionary Change for Your Technology Business. *Most recently, he cofounded the first Limited WIP Society in Germany. Mr. Roock can be reached at arne.roock@it-agile.de.*

Demystifying Kanban

by Alan Shalloway

Ask the question "What is Kanban for software development?" and you are likely to get many different answers. It is a powerful new method that addresses challenges previous methods have not. It is a transition management system. It is a more effective method for teams to deliver business value incrementally. It is a way to create visibility for executives to improve product portfolio management. It is all of those things and more. It is like that *Saturday Night Live* spoof in which Gilda Radner and Dan Aykroyd argue over whether New Shimmer is a floor wax or a dessert topping; they finally decide it's both!

In this article, I describe Kanban as a systems approach to software development that affects many types of behaviors. I also mention a few of the common misconceptions people have about Kanban in order to help clarify what Kanban is and is not.

ITERATIVE AGILE

Agile software development has been embraced by many in the software industry. First-generation agile methods, such as Scrum and XP, took an incremental approach based on well-defined, timeboxed iterations (which Scrum calls "sprints"). The team commits to building a certain number of product features they feel they can reasonably accomplish within that increment. At the end, they see how they performed so that they can refine their commitments for the next timebox.

The advantages of an incremental approach include:

- Promoting an understanding of the need for prioritization
- Requiring the team to finish the work they have started
- Protecting the team from interruptions and distractions
- Defining regular intervals for the business to adjust priorities in order to ensure the team is working on the right things
- Defining regular intervals both to demonstrate what has been built and to gain user feedback while it is still useful

These agile methods require cross-functional teams whose members focus only on their team's work. A cross-functional team is one that contains most or all of the resources needed to perform the activities required to deliver product: analysis, design, coding, and testing. If you can afford it, cross-functional teams are indeed powerful; in fact, you may gain as much value merely from employing cross-functional teams as you would from using an agile method.[1, 2] The reality, however, is that many organizations cannot afford wide-scale deployment of exclusive cross-functional teams. This is an impediment to scaling these methods to the enterprise.

Another impediment is that team-based agile methods do not effectively address one of the most common problems in development: more things being pushed onto teams than they can handle. Team-based approaches start at the wrong end, attempting to insulate or protect the team from intrusions. This fails to address overall capacity, has the side effect of isolating teams from management, and actually works against scaling.[3] The better approach is to provide visibility to those generating work — executives and management — so they can see the impact of overloading teams. They need to see this both at an individual team level and, more important, for the entire value stream.[4] Creating agility at scale requires an approach that handles the entire value stream.

ABOUT KANBAN

Based on lean principles and the theory of constraints, Kanban is a second-generation agile approach that addresses the entire value stream and overcomes the challenges inherent in team-based agile approaches such as Scrum and XP. The motivations behind Kanban include:[5]

- **Controlling the rate of transition.** First-generation methods often require traumatic change to the people and structures in the organization.
- **Allocating specialized skill sets, domain knowledge, or knowledge of legacy code effectively across the organization.** Dedicated and relatively static teams, while desirable, are usually not practical in this regard.
- **Enabling participation of management and leadership.** Scrum often marginalizes management.
- **Providing teams with guiding principles,** especially the principles of lean and the theory of constraints.
- **Providing a better way to learn how to improve.** End-of-iteration retrospectives are too narrow and too late to be valuable over the long haul.
- **Enabling teams to work on the right-sized chunks.** With timeboxing, work gets squeezed into a predefined period and unrelated bits of work get grouped into one sprint.

In this section, I will describe how Kanban acts as:

- A lean-agile team method
- A transition management system
- A means of learning
- An aid to product portfolio management

Kanban as a Lean-Agile Team Method

One of the premises of Kanban is that, to be efficient, teams must not become overloaded. Overloading teams causes thrashing, creates waste, lowers quality, and harms capacity. Getting to a sustainable load requires focusing on how many things a person is working on at any one time: too few results in ineffective loading of the team; too many results in additional work caused by delays.

Timeboxing does help with this because the team limits the amount of work they will allow into the iteration. The problem has been made small enough for the team to handle — however, it is not helpful to anyone else. Everyone else must adjust work and priorities to accommodate the team.

Kanban has a different focus. It is based on a commitment to optimizing the flow of work across the entire value stream: the work that is done from when an idea is initiated until it is consumed by the customer (internal or external). This broader perspective enables a business-driven approach. Kanban's methods provide for team management within the context of the broader value stream. It requires participants across the value stream to understand the lean tenet of avoiding too much work in progress (WIP) and the techniques for doing that. Kanban assumes that improvements are desired beyond just individual team performance. What is Kanban's method for avoiding too much WIP? It takes the following approach:[6]

1. Make all work visible.
2. Have management and the team agree on a set of goals. This involves those upstream of the team, the team itself, and those downstream.
3. Define where the Kanban method will be applied. For example, Kanban addresses the team's work from the point at which they consider items on their backlog until they deliver that work to the customer.
4. Create a board that reflects the team's flow of work.
5. Educate the team on the principles of flow and pull.
6. Start managing the work with pull methods and set WIP limits that maximize flow through the system.
7. Define explicit policies that control the flow of work and continuously improve them.

Here is one more issue to consider under the topic of team management: managing the various points of interface with the business. There are at least four:

1. When the business gives work input to the team
2. When the state of development is evaluated
3. When features are demonstrated to stakeholders
4. When the team and the business commit to what will be done next

Kanban does not require timeboxing and therefore decouples these event points by allowing them to happen at different times. However, most Kanban teams have found it useful to update and review Kanban backlogs at regular intervals. This is called the "cadence" of the team. Be clear, however, that this is not the same as a timebox.

MISCONCEPTION: KANBAN IS SCRUM WITHOUT ITERATIONS

Kanban can actually be done with iterations if there is a business or team reason to do so. Teams that have little discipline in finishing things may find the hard stop of the iteration to be useful. But most find them an unnecessary interruption and a cause of extra work.

Kanban as a Transition Management System

In the best of circumstances, change is hard. The most successful change efforts involve explicitly managing the transition. This lowers resistance and increases innovation and results.

Scrum can present challenges in transition. Right from the start, it demands new organizational structures (e.g., cross-functional teams) and requires people to assume new roles and a new set of practices, while providing them little guidance. This can be traumatic to the organization and certainly causes resistance.

Kanban takes a different approach to transition. You start where you are and incrementally improve the bottlenecks in the whole value stream, guided by explicit policies and data. This is an important distinction. Except in the case where cross-functional teams are easy to form and people are happy with change, a more transition-friendly approach is required. In other words, if the proposed change is more than the people can cope with, a way of improving throughput without a dramatic team organizational shift will be needed.

In Kanban, the goal is to improve work by removing the bottlenecks in your workflow. Where there are issues, management and the team can do load balancing by changing how they are doing the work or who is doing the work. This makes transition a series of easier steps along the journey toward optimal flow across the value stream. It is much easier on the organization.

MISCONCEPTION: KANBAN SUGGESTS LINEAR WORK AND REQUIRES TOO MANY HANDOFFS

Actually, Kanban does nothing of the sort. It suggests that you start where you are. If you have too many handoffs, the Kanban board will make this clear. Kanban believes changes made by the team are best made when the team understands the cost of current practices, decides what to do, and then sees the results of their work.

MISCONCEPTION: EXPLICIT POLICIES ARE STATIC, DETERMINISTIC, AND HARD TO CHANGE

None of these is true. One can have a policy that states: "Our policy is that you do what you want, when you want to do it." That's explicit, not static, not deterministic, and not hard to change. It also isn't very useful. Nevertheless, it proves the point that "explicit" does not mean any of these things. It simply means that everyone has talked things through and understands what the policies are.

MISCONCEPTION: KANBAN IS NOT "PEOPLE FRIENDLY"

Talking about people does not make a method people friendly. Actually *helping* people makes a method people friendly. Kanban is a systems thinking approach to solving the problems people have — which is very people friendly.

Kanban as a Means of Learning

Kanban addresses the maxim "People know what to do; they don't always do what they know." By creating clarity on the effects of their actions, Kanban encourages people to do what they know.

This approach to improvement requires high-quality conversations between the various stakeholders, developers, and leaders. They need to remember what has been agreed to (the current basis), know the facts about what is happening (the current situation), and be clear about what will change (the new basis). Kanban requires both the work itself and how the work is done to be clear to everyone. This means using explicit process policies to define the basis of work. Improvement happens by:

- Identifying problem areas by measuring queues on the Kanban board
- Developing a new or revised policy or changing the current WIP limit
- Seeing what the effect these steps have on smoothing out queues

Kanban focuses the team's energies on improving those areas in the value stream that are truly constraining other work (and thus, harming flow through the value stream). It helps avoid wasting time on activities that do not really help flow. By having a before, during, and after action method of making small changes, the team learns what works and what doesn't.

Here is an example. Say there is a bottleneck in getting things tested. With Kanban, the queue of work in front of the testing team reaches a limit so it cannot receive any more work. Developers are restricted from doing more coding. This is good, because coding more in this situation would be counterproductive — it would just increase the time between coding and testing even more. Looking at the queues, it becomes obvious that the team needs to determine how to get testing more in step with coding. Perhaps the team will decide to do coding and testing together;[7] perhaps they will get more testers or offload other work from the testers so they can concentrate on testing. Kanban does not specify the solution, but it does identify the problem and the result needed to solve it (i.e., reducing the queues).

Kanban's learning method is therefore integrated into its management method. This is both its power and why it is sometimes not fully appreciated. In team-based agile methods, much of the learning occurs at the end-of-iteration retrospectives. These retrospectives produce useful results early on but then grow stale as changes produce less dramatic results. What to do? You could tell teams just to keep at it, to limit the number of changes in each sprint, or to learn how to run retrospectives more effectively. But these don't attack the real problem: focusing too much on the local team and not the larger value stream.

The problem is that learning at set stages is not nearly as valuable as learning continually. Developers are an interesting lot. One of their bigger strengths is their passion. Yet it is just this passion that often prevents them from stepping back and seeing if there are better ways to do things (hence, the prescribed Scrum retrospective). Through the use of the Kanban board, which makes visible the consequences of the team's decisions, Kanban provides a way to learn continually. Team members no longer need to "step back" to learn.

The use of explicit policies in Kanban helps people validate their understanding about what to do. And it enables both teams and management to see cause and effect whenever they make a change; thus small adjustments lead to quick results.[8]

Kanban as an Aid to Product Portfolio Management

One of the most chronic and severe problems facing software development today is that teams are overloaded with work. Working on multiple things obviously delays the delivery of any one particular item, but what is not as obvious is that the delays between the different work steps (e.g., writing a story and coding it, or coding a story and testing it) can actually increase the amount of work to be done. This is because work often has to be redone (as in the case of requirements) or what was in the short-term memory of the developers has to be relearned in order to complete the task (as in the case of debugging).[9]

Managers become frustrated when jammed development teams fall behind and will actually demand that they do more — creating even bigger jams. To break this cycle, it is not sufficient to simply improve a team; you must help management understand the impact of their demands. Through the Kanban board, Kanban offers explicit tools that provide visibility and help management make appropriate decisions about tasks based on business value. The result is that teams are less overworked and more focused on the right things.

One of the most chronic and severe problems facing software development today is that teams are overloaded with work.

In Kanban's flow model, in which team members just pull things off the backlog whenever they are ready, new items coming in can be worked on very quickly. Unfortunately, timeboxed methods require teams to wait until the beginning of the next iteration to take on something new or unexpected, things that are all too common: Severity 1 errors, demands from an irate client, an executive's "good idea," and so on. Scrum lacks an explicit way to handle these situations. While it would like to protect the team from such interruptions, in reality the team gets pressured to add in the extra work, but they have no way to demonstrate the impacts of this on management.

In his book *Kanban: Successful Evolutionary Change for Your Technology Business*, David J. Anderson offers a great example of how visibility can help managers make good decisions. In the example, a team agreed that product managers could ask team members to work on an urgent item even if it caused them to exceed their WIP limit. However, they would not be asked to work on more than one urgent item at any one time. Their history of managing via WIP limits had demonstrated that going beyond them would slow the team down, but they also recognized that at times an urgent situation demands that.

Sure enough, the day arrived when a manager played the "expedite" card. Interestingly enough, David did not challenge it. If it made business sense to get something out the door at the expense of everything else, so be it — that was what the team needed to do, and it was a product manager's decision. But someone else *should* challenge the decision, and someone *did.* That "someone" was the other product managers. They demanded that the first product manager explain why expediting his work at the expense of others' was a good decision for all involved. By creating visibility into the team's process, Kanban clarified the impact of adding "just one more thing" to the team's load. *All* the product managers could see the impact of what, and how much, the team was being asked to do. In this way, Kanban helps organizations avoid the local optimizations that team-focused methods tend to inadvertently encourage.

While it is true that Kanban requires a certain understanding of lean principles, it doesn't require organizations to learn prescribed practices and roles to get started.

SOME COMMON MISCONCEPTIONS ABOUT KANBAN

Let me close with a few comments about some other Kanban misconceptions:

- **Kanban has been successful because it is being done by early adopters.** Actually, Kanban is used by several different camps. First, there are the early adopters, who have created and refined it. They created Kanban software development to benefit the other two camps, the first of which is organizations that found other agile methods difficult to apply. The second camp consists of those who were late adopters of Scrum because Scrum required too much change to adopt. In addition, the success of Kanban has led many organizations new to agile to adopt it directly.
- **It is better to start with Scrum and then move to Kanban.** This idea originated with the Scrum community, and I have always considered it a bit self-serving. While it is true that Kanban requires a certain understanding of lean principles, it doesn't require organizations to learn prescribed practices and roles to get started. Many teams do start with Scrum and then move to Kanban, but typically only because they were not aware of Kanban at the beginning.
- **Kanban is mostly good for support.** Many practitioners started using Kanban in support environments where it didn't make sense to batch up small pieces of work to create sprints. However, other teams working on new projects with more sizable features have found Kanban works equally well regardless of feature size. In the same way that Kanban provides a choice of using iterations or not, it provides a choice of whether to break features down into smaller pieces. Experienced teams have found it useful to work on smaller stories in order to get quicker feedback on both design considerations and from the customer. Where breaking features down does not make sense, however, Kanban does not demand it. This is another example of the way Kanban enables teams to decide how fast they change their practices.

IS IT KANBAN? PUT IT TO THE TEST

Perhaps the best way to define Kanban is to define a test that indicates whether you are doing it or not. A team can be considered to be doing Kanban if do the following:

- Make all of their work visible.
- Limit their work in progress to their capacity.
- Manage their workflow in order to improve cycle time (the time it takes from starting work on the feature until it is completed).
- Make all process policies explicit.
- Improve collaboratively and continuously based on facts and explicit policies.

Notice that the proof you are doing Kanban does not lie in following certain prescribed practices, but rather in attending to particular elements of your work.

TO RECAP

Kanban is designed to overcome the three largest impediments to agile adoption beyond the team level. These are:

1. The trauma often caused by creating cross-functional teams as required by other agile methods.
2. Too many feature requests hitting the teams, causing them to thrash.
3. Lack of sustained learning after initial adoption.

It addresses these problems by applying solid lean principles, including:

- Reduce delays by limiting work in progress to the capacity of the team.
- Create visibility in both the work being done and the way the work is being done.
- Include management in the transition.
- Explicitly incorporate learning and knowledge stewardship.

Kanban is proving itself highly effective in many contexts. Many XP- and Scrum-based teams have found it to be helpful in overcoming challenges they have had, challenges that they might *not* have had if they had just started with Kanban.

ENDNOTES

[1]Shalloway, Alan. "Challenging Why (Not If) Scrum Works." NetObjectives, 24 May 2007 (www.netobjectives.com/blogs/challenging-why-not-if-scrum-works).

[2]Shalloway, Alan. "How Successful Pilots Actually Often Hurt an Organization." NetObjectives, 3 October 2010 (www.netobjectives.com/blogs/how-successful-pilots-can-hurt-the-organization).

[3]Shalloway. See 2.

[4]A "value stream" is the set of actions required to develop the software and includes how it is conceived, developed, and deployed.

[5]For more about the differences between Kanban and Scrum, see: Shalloway, Alan. "The Real Differences between Kanban and Scrum." NetObjectives, 7 June 2010 (www.netobjectives.com/blogs/real-difference-between-kanban-scrum).

[6]This is a gross simplification of the starting Kanban process as outlined in Anderson, David J. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

[7]XP has long advocated this approach, called Test-Driven Development (TDD). Several of TDD's thought leaders consider it a manifestation of Kanban thinking.

[8]While I do not believe the best place to learn lean is from Toyota, I highly recommend Rother, Mike. *Toyota Kata: Managing People for Improvement, Adaptiveness, and Superior Results*. McGraw-Hill, 2009.

[9]Readers who are interested in this well-known but rarely discussed phenomenon of software development — I call it "induced work" — should watch a short video titled "How Delays Cause Waste: A Main Tenet of Lean" at www.youtube.com/watch?v=5S97z0taHB8.

Alan Shalloway is the founder and CEO of Net Objectives. With more than 40 years' experience, Mr. Shalloway is an industry thought leader in lean, Kanban, Scrum, and design patterns. He helps companies transition to lean and agile methods enterprise-wide and teaches courses in these areas. Mr. Shalloway has developed training and coaching methods for lean-agile that have helped his clients achieve long-term, sustainable productivity gains. He is the primary author of Design Patterns Explained: A New Perspective on Object-Oriented Design, Lean-Agile Pocket Guide for Scrum Teams, *and* Lean-Agile Software Development: Achieving Enterprise Agility *and is currently writing* Essential Skills for the Agile Developer. *Mr. Shalloway is a popular speaker at prestigious conferences worldwide and a cofounder and board member of the Lean Software and Systems Consortium. Mr. Shalloway has a master's degree in computer science from MIT and a master's degree in mathematics from Emory University. He can be reached at alshall@netobjectives.com.*

Kanban at an Insurance Company in the Netherlands

by Dan Verweij and Olav Maassen

ADOPTING KANBAN AT ASR INSURANCE: THE TIMELINE

ASR Insurance is one of the top three insurance companies in the Netherlands. A couple of years ago, ASR adopted an agile method, Scrum, as a way of working for IT projects, and it proved to be very effective in delivering project results. Based on this outcome, Scrum was implemented widely across the company in IT projects and maintenance and operations (M&O).

While effective as a chosen methodology for projects, Scrum didn't work well as the default way of working in an M&O environment. Maintenance and operations often have to work on items with a short time frame and be able to respond quickly. Not infrequently, this turned out to be at odds with the planning of sprints.

In August 2009, ASR launched a pilot with a team of around 10 people. They were tasked with continuing their maintenance work while switching to Kanban as their new way of working. The first results were so encouraging that more teams started to work with Kanban. In October 2009, ASR had seven teams running with Kanban. Some of the teams were mixed M&O teams, while some others were dedicated to only maintenance or only operations. The experiences of these teams gave ASR the necessary confidence to take the next step in Kanban adoption.

At the same time, three other forces were active within ASR:

1. **Dedicated maintenance and operations teams.** To service the rest of the company better, the IT department wanted dedicated teams for maintenance and operations, respectively. One team would be responsible for doing maintenance on an application (i.e., changes to the functionality) and another team would do the daily operations for that application (i.e., making sure it runs to spec every day). Each team would be dedicated to a set of applications, and the team members would be 100% allocated to their team.
2. **Matrix organization.** To enable the dedicated maintenance and operations teams, the IT department wanted to transition from a more traditional hierarchical structure toward a new matrix organization. Based on previous results, Scrum would be the default method for projects and Kanban the default method for M&O.
3. **Lean/operational excellence program.** At the board of directors level, a lean and operational excellence (OpEx) program was started to help improve the efficiency of the whole company. The OpEx program introduced lean working within all the regions of the company, including internal certification.

In January 2010, nine new teams for operations were formed and started working with Kanban. Three months later, nine other teams were formed to handle the maintenance for applications. Introducing this change was quite intrusive and was coached by both the lean program and the agile adoption program to provide maximum support. The lean program focused mostly on the continuous improvement of the new way of working, and the agile adoption program with Kanban focused on work management. Both programs helped in areas like cultural change, management, leadership, behavior, and personal responsibility.

As of February 2011, 18 teams are working with Kanban within maintenance and operations, with 160 people divided equally between the two areas. Each multidisciplinary team contains developers, architects, business representatives, information management experts, and whatever other skills and roles are required to do the job. Each team is facilitated (not managed) by a team coordinator to keep the process running smoothly.

REASONS FOR ADOPTING KANBAN

In today's economic climate, efficiency and effectiveness are important. At ASR, the focus on cutting costs and improving efficiency resulted in a cultural change and a real sense of urgency, which is needed for successfully implementing change in an organization.

The most important reason for implementing Kanban at ASR is *alignment with the business sectors*. Because the budget for operations is fixed (i.e., by having a dedicated team working on it all year), the internal IT clients know beforehand the cost of keeping applications running. This increases their desire to be more closely involved in decision making to make sure the team is working on the most important things all the time. The fixed budget serves to cut costs, while Kanban allows the clients the flexibility they desire within that budget.

The second reason for adopting Kanban is the tremendous *transparency* it creates. In the past, IT was seen as a pit into which one threw money and hoped for the best results. Because the clients are now closely involved, they gain more influence and have a bigger say in what happens. Transparency combined with this added influence greatly increases the trust between all parties involved.

When ASR started working with maintenance and operations teams, they first used Scrum because of the company's positive experiences with it in projects. However, the short cyclic nature of maintenance and operations and the often isolated tasks made Scrum less than ideal. Although Scrum also has a short cyclic nature, it became impossible for the M&O teams to commit to sprints, as they were unable to forecast what they would be working on for the next two weeks. Any request coming in could be of a higher priority than anything the teams were working on at the time, and as a result there were many failed sprints.

We have seen teams in which the individuals were all so focused on their own work packages that a stand-up meeting didn't provide them any information. There was no cooperation, and people were staring out the window. The visual management aspect provided by things like the Scrum board and the impediment list, however, proved very valuable. Kanban also has a very strong *visual management component* that allows teams a much better overview, and concepts such as flow and pull make it more suited to the short cyclic nature of M&O work. By changing their way of working from Scrum to Kanban, team members started to cooperate and help each other, just as a team should. Kanban visually demonstrated their dependence on each other. For example, when Kanban showed that testing was the bottleneck for the team's throughput, the developers decided they needed to focus on helping the testers do their work. Further investigation (and experimentation) later on showed that many of the issues found during testing could be prevented if the testers and developers cooperated more closely when the team started developing each feature.

In software development and software engineering, the agile way of working is the best possible implementation of lean principles in IT. Lean and agile initiatives reinforce each other. Principles such as flow, waste reduction, and continual improvement are natural parts of agile. Lean helps in connecting the agile mindset to the business.

By changing their way of working from Scrum to Kanban, team members started to cooperate and help each other, just as a team should.

BENEFITS OF IMPLEMENTING KANBAN

Now we know why ASR chose to implement Kanban. What other benefits have we seen?

Improved Quality and Increased Production

Based on early results, we can see that the number of change requests solved and implemented by the teams has doubled regardless of the size of the change requests. The number of incidents (problems in production) has decreased, and the number of resolved incidents has risen with the early teams. We see comparable progress with most teams where Kanban is implemented and running well.

Improved Business-IT Alignment

The business is heavily involved with the Kanban teams. The transparency, fine-grained control, and consistent delivery over time have restored the trust between business and IT. Real partnerships have developed between teams and the business departments they work for. The team that achieved the best results in improved quality and production was the team that was located in the same hallway as the business department they served. This physical proximity increased their involvement with each other.

The business departments now feel so involved with the teams that it is quite natural to see business people at the stand-up meetings in the morning. Even when nobody from IT is present, the business representatives start up the electronic screen to display the Kanban board containing all the work in progress, and *they* hold a stand-up meeting. This shows that they now feel part of the team.

Better Information Flow

Having the business units present at the team locations creates more understanding of each other. Business people are responsible as part of the team and can now directly see what is keeping the team from reaching intended goals and can actively help resolve issues. When things turn out to be more difficult than initially estimated and take more time, it is more logical to the business because they receive more information directly from the source.

This also works the other way around. The information manager can explain why a particular change request is very important and that there are many people waiting for the change. The turnaround is much quicker, more flexible, and easier to manage than going through the official issue-tracking systems that ASR had been using. Personal contact has replaced communication through tooling. The issue-tracking systems are still used, but now they are just for tracking issues instead of for communication.

The danger is that the term "Kanban" becomes more widely used, but without an understanding of the fundamental principles and concepts that back it up.

Simpler Organizational Structure

The way Kanban was implemented at ASR (i.e., with dedicated teams) allowed the company to create a simpler organizational structure. Many Application Services Library (ASL) roles previously instituted became obsolete, and the responsibilities were transferred to the teams. ASR no longer has people in the specific roles of incident manager and incident coordinator, for instance. The team handles all incident management because everybody can see the status. (The ASL roles that still remain separate are knowledge manager and quality manager.)

Cost Savings

Although cost cutting is not the primary goal of implementing Kanban, costs for maintenance and operations at ASR have been lowered. This is a consequence of the increased transparency. There is a constant focus on doing the right things with the people available in the M&O teams. This increased effectiveness reduces the number of people needed to do work on maintenance and operations. The close relationship between the business departments and the teams reduces much of the previously needed overhead.

DIFFICULTIES OF IMPLEMENTING KANBAN

Although the benefits of implementing Kanban are clear, there were challenges in getting Kanban adopted at ASR. This section highlights the most important difficulties encountered during the Kanban implementation.

- **Personal efficiency mindset.** The biggest step people have to take is to change their mindset. Society is very focused on efficiency, and people want to contribute all the time. This focus on personal efficiency, however, tends to lead to a suboptimization of the whole team. Sometimes it makes more sense for a team member *not* to pick up a task that he or she is an expert at and first help a colleague so the team can deliver results. What Kanban asks of teams is to focus on the end result and make their decisions based on effectiveness. This affects such decisions as what work the team picks up first and what tasks the team puts the most effort into. The teams become responsible for the results, and the individuals are responsible to their team. It takes time for people to change their behavior from something they have been doing for many years (i.e., focusing on personal efficiency) to a more collaborative mindset.
- **Ch-ch-ch-ch-changes.** Complicating matters at ASR was the fact that many changes were implemented at the same time. Besides the adoption of Kanban, people were also facing a new location, a new boss/manager, a new matrix organization with two managers instead of one, and a new office setup — employees no longer had their own desks because of a Results Only Work Environment.[1] The whole mix of changes at the same time hindered the speed of the Kanban adoption.
- **Monkey see, monkey do.** Despite the many changes in the environment, teams eventually became successful with the use of Kanban. Teams that had been working with Kanban before these changes were implemented had a head start and were up and running sooner than the other teams. The success of the early Kanban teams was soon noticed, and teams that weren't part of the Kanban adoption at that time wanted to get similar results. So they copied what they could see — namely, the Kanban board and the stand-up meetings, which are the most visible elements from the outside. Because these teams missed some of the essential ideas behind Kanban — such as flow, pull, and continuous improvement — they were able to improve, but not to the level of the other Kanban teams. The danger is that the term

"Kanban" becomes more widely used, but without an understanding of the fundamental principles and concepts that back it up.

- **You say "potato," I say "po-tah-to."** Being in a non-English-speaking country, the team responsible for the agile (and Kanban) adoption decided not to translate the agile and Kanban terminology but to keep the English terminology. The benefit is that there is more information available on the Internet in English, and it becomes easier to relate the theory to the practice. In some parts of the company, however, certain standard terms such as "stand-up meeting" ("Dag start") and "retrospective" ("Keek op de week") were translated. We observed that this led to confusion about what the terms exactly meant — especially in uncertain times, as during an implementation of a new way of working, people are looking for certainty in definitions. Having different words for the same thing didn't help.
- **Lack of business involvement.** One important challenge was getting the business actively involved in what the teams were doing. It didn't help that initially the teams and the business were located at two different locations 15 miles apart. This separation reinforced the divide between the business and IT. At first, the business was represented at the teams' location one day a week. It took quite some effort to get the business really involved with the IT teams.

WHY NOT KANBAN EVERYWHERE?

Just as ASR considered using Scrum for all things related to IT, the company also debated whether Kanban could be used for their development projects. And although Kanban works well for maintenance and operations, running a development project using Kanban at ASR doesn't seem to be a good fit. We expect Scrum to be used when delivery in batches is desired. This makes more sense than using Kanban.

In one case, ASR started a large-scale maintenance project for one of their applications. At least the *original* request was for maintenance, but it turned out the assignment was so big that it needed to be organized as a project instead. The project started with Kanban as a way of working under the guidance of an experienced agile coach who had successfully coached both Scrum and Kanban teams within ASR before. Although all the conditions were positive, the team was not able to produce the desired results while working with Kanban. Management then decided that the team should switch to Scrum, and that method worked much better for the project.

Although experience with agile methods is not a prerequisite for a successful Kanban implementation, it does appear that it helps to get Kanban working within projects more quickly. Because the team lacked such experience, they needed more structure to be able to perform. Using Scrum and having two-week iterations without scope changes helped the team get better results. We expect that as their experience in agile grows and their search for continuous improvement intensifies, many Scrum teams will move toward Kanban.

If a project has clear requirements, interactions are clear, and the project organization is temporary, it makes sense to stay with the currently chosen way of working instead of introducing a new methodology such as Kanban. Introducing a new concept has a cost associated with it, and that might be more than the benefits Kanban would bring to that situation. Kanban is great in environments where there is a high degree of interaction with the clients or in very innovative environments where requests and feedback are constantly pouring into the team.

RECOMMENDATIONS FOR ADOPTING KANBAN

If you are considering adopting Kanban as a way of working, we offer the following recommendations based on our experience with a large-scale Kanban adoption:

- **Involve people and prepare them well**. Changing the way people work together means organizational change. For this to be effective, it is essential to explain why all this change is happening. Clarify the problems you see and what you think should be done about them. Before going forward and implementing Kanban, allow people to make the same mental journey that you did. Create a platform for analysis and discussion in order to let people become convinced of what they need to do. Follow that with training on Kanban.
- **Allow for a transition period**. People will not be convinced just because you tell them to be. Many people working in IT are very analytical and need some form of proof that this method will work. Take the time to create and foster support among those who are most affected. After the majority has been convinced, execute the transition to convince the rest. Take away

any further doubts by enabling them to experience the benefits of Kanban for themselves. Have them start with Kanban in a team and see the benefits of visual work management and better cooperation between team members and other stakeholders. These benefits are felt quickly after starting, and they help people to accept Kanban as the new way of working.

- **Think long term**. There is a demand to realize ROI quickly. Managers tend to focus on quick wins, and this makes it harder to convince people operating within the change. Kanban adoption takes many small steps, and when considered in the short term, it may not always look efficient. The power of Kanban lies in its focus on increasing the effectiveness of the whole system. Each of the steps might be small, but the total effect is big. Looking back over the last two years at ASR, a lot has been realized by thinking long term.
- **Ensure close proximity**. Have the teams and the people they work with sit close together. This helps the organization create and maintain the necessary transparency. It removes many of the impediments that keep teams from performing and fosters understanding.

CONCLUSION

A couple of years ago, ASR Insurance adopted agile as a way of working for IT projects. Recently ASR has extended that choice to maintenance and operations. Scrum has proven to be a very effective way of delivering results in projects, but for maintenance and operations ASR prefers Kanban. After working with Kanban, the company has found that quality and production have improved, and there is better alignment between IT and business. Both now have a shared focus on doing the right things. Although ASR experienced some difficulties in adopting Kanban, the benefits easily outweigh them.

ENDNOTE

[1]See http://en.wikipedia.org/wiki/ROWE.

Dan Verweij is currently Program Manager Operational Excellence ICT for ASR Nederland. He is responsible for implementing lean and agile work processes in the ICT environment. It is his belief that agile methods are the most "lean" way of developing software. Previously, Mr. Verweij was manager of a large department of developers at ASR. It was during this period that many of the developers were first introduced to agile methods such as XP, Scrum, and Kanban. Mr. Verweij can be reached at dan.verweij@asr.nl.

Olav Maassen is a Principal Consultant for Xebia in the Netherlands and was Kanban coach during a large part of the Kanban adoption described in this article. Mr. Maassen's central theme is to help others perform at their best. He is interested in new developments and new ideas that can help IT professionals improve themselves. He knows how to inspire people to optimize their capacities/skills in order to get the best results for both themselves and the company.

Mr. Maassen is an active participant in the agile community. He was one of the founders of AgileHolland, a foundation for promoting the use of agile methods in the Netherlands, and he is actively involved in organizing international conferences. Furthermore, Mr. Maassen is often invited as a speaker and presenter at international conferences. The recurring theme of (most of) his presentations is how transparency and trust can influence an organization. Mr. Maassen can be reached at olav.maassen@xebia.com.

Kanban for Help Desks: Managing the Unplannable

by Roland Cuellar

Many organizations have a service desk or help desk that is set up specifically to take all of the unplanned, ad hoc service requests that come about as a normal function of daily operations. These are often rather small matters, such as minor software defects, perplexing hardware issues, printer problems, phone issues, and so on. The list is practically endless. But sometimes these service requests can be major issues that have significant external customer-facing impact as well — networks that are down, major customer-facing applications that are down, serious hardware outages, and the like.

One of the more interesting aspects of this work is that it is, by its nature, unplanned. We don't know what kinds of issues will arise, when they will come about, how severe they will be, or what skills will be required to address them. Add to this that the help desk function must deal with hundreds or thousands of customers, all irate, who see their "high-priority" issues go into some black-box ticketing system, often with no real feedback on when their issue will be resolved, and you have a recipe for 360-degree frustration. For a customer, the only way to ensure good response time, then, is to make your ticket a "critical" one. This, of course, leads to the help desk being bombarded with lots of so-called critical issues. On top of all of this, remember that most help desks are cost centers that must operate with minimal funding, which results in a chronic shortage of staff and other resources. These myriad challenges can make managing help desks a maddeningly complex endeavor and a seemingly intractable problem. How can you manage the unplannable?

TRADITIONAL APPROACHES

Many help desks operate within the context of larger organizations, where process is king. Mature organizations have a detailed process for just about everything, so there is frequently an attempt to try to force the help desk function to fit one of the existing internal project development processes — often with less than encouraging results. Let's consider two common approaches — waterfall and Scrum — and some of the typical problems that result from their use.

Waterfall

In a waterfall process, there is a set of fixed development phases, and the object is to move a fairly large batch of requests collectively through the phases as a group. In this model, we would take a bunch of high-priority bugs and small enhancements and perform some level of requirements analysis, followed by design, development, and testing. In my experience, this model rarely works well even for planned software development projects. In the help desk world, the constantly changing priorities make it even more difficult to succeed with this model. Almost as soon as you try to lock the scope for the next release, some new high-priority issue will arise, forcing you go back to the beginning of the cycle.

The waterfall process is predicated on stable requirements. It basically says, "If we can lock down the scope and not let the requirements change, then (and only then) we can perform detailed analysis, design, development, and testing in a single pass. Any change in midstream forces us to go back to the beginning." In short, this process is not designed to deal with ever-changing priorities and requirements. Even if you could hold the line on requirements changes, the inherently long delivery cycle that comes from large batches of requirements all going the through the process together is not a good fit for the fast-turnaround expectations of a support environment. The waterfall approach is simply not agile enough for the realities of the help desk.

Scrum

Forward-thinking organizations that have already adopted an agile method such as Scrum will often attempt to use it to manage help desk development. Scrum's smaller batch sizes, short delivery cycles, and requirements independence[1] all tend to make the method a much better fit for the high-variability world of the help desk. However, many of the Scrum implementations I have seen for managing help desk work have been problematic. Scrum has high expectations that are difficult to uphold in a highly reactive environment. For example, Scrum expects the team to have a very clear set of priorities going into an iteration and to lock the scope, accepting few if any changes during a sprint.

So while Scrum is a much better fit than waterfall for managing help desk development, it is still at odds with the reality of the situation. If the team is in the middle of a sprint and a showstopping defect surfaces, then the team is going to have to abandon much of their current work in order to deal with the crisis. And if you are operating in an environment where there is a new crisis just about every day, then even a highly agile Scrum team may not be agile enough to handle this level of continuous change.

No traditional plan would survive even a few hours in this environment.

Now, some teams can deal with this by holding back some capacity during sprint planning for any unplanned, high-priority work that may need to be addressed during the sprint. This can work fine if the number of late changes is relatively small and manageable. But when teams are holding back 40%, 50%, or more of their capacity in order to deal with unplanned work, it is clearly an indication that the process is not a good fit for the situation.

Pitfalls of Both Methods

The basic problem in both waterfall and Scrum is batch size; in each approach, the batch size is simply too big to deal with continual change. From queuing theory and inventory management, Little's Law[2] tells us that batch size is proportional to cycle time. The larger the number of items in the processing batch, the longer it will take to process them all, so the longer the delivery cycle will be. Any process that is going to work in this environment will need to have a very small batch size in order to respond to continual change and achieve the fast delivery/response time that is expected in a maintenance environment. In fact, for a process to be successful in this environment:

- The process must be open to almost continuous requirements reprioritization.
- Items must be able to progress through the process independently of one another.
- Batch size must be small.

I have experimented with Kanban in several client organizations as a possible solution to this problem. One organization is a huge *Fortune* 500 firm and has all of the complexity that one typically sees at that scale of operations. Another is a small independent media firm that is at the complete other end of the spectrum in terms of scale and complexity. Experiments with Kanban in both of these help desk organizations have yielded very positive results.

KANBAN FOR SERVICE AND HELP DESKS

The Situation

I'll use the smaller client (let's call it "DynamicMedia") as an example, but the basic challenges at both clients were fairly similar. At DynamicMedia, there are two service teams, the help desk team and the network operations team, each of which works primarily in a support function. And each of the two teams has two kinds of work: some planned projects and a lot of unplanned support calls.

The help desk takes support calls for a huge variety of internal issues, everything from software defects to printers that need new toner cartridges and e-mail and PC problems. The networking team supports calls related to the LAN, back-office hardware infrastructure, storage, phone systems, and software infrastructure. Planning and managing in these environments is extremely challenging due to the constantly changing issues, help requests, fluctuating priorities, and lots of simultaneous efforts. No traditional plan would survive even a few hours in this environment.

In addition to the constant requests for support, these two teams have some traditional project work to deliver. Project examples might be e-mail system upgrades, network upgrades, or new hardware installs. The project work needs to be estimated, budgeted, planned, and delivered as with any project. But trying to deliver on project plans for the planned project work is highly difficult in this environment due to the interruptions coming from the constant support issues.

My colleagues and I had already helped the software development teams at DynamicMedia move to agile/Scrum software development processes, and the management team desired a similar approach for managing the support teams.

Approaches

In a typical software development environment, there is often a lot of planned development work combined with some unplanned support work. Here the situation was reversed: a lot of unplannable support work with some planned project work. Scrum with its timeboxed iterations didn't seem like a good fit. Help desk calls, network outages, and the like don't lend themselves naturally to well-delineated, fixed-scope Scrum iterations, and a two-week delivery cycle on a network outage is nonsensical. So while we probably could have made Scrum work, we decided to take a Kanban approach instead.

WHAT IS KANBAN?

In a traditional waterfall process, we use large batches and huge amounts of work in progress (WIP) as the planning paradigm. We do all of the requirements elaboration, then all of the design, then all of the coding, then all of the testing. These huge batches typically take months to deliver in even the best organizations.

Scrum uses much smaller-sized WIP buckets. In Scrum, we break the transfer batch size down to two-week chunks. We do a little bit of requirements analysis, a little bit of design, a little bit of development, and a little bit of testing in order to deliver a handful of features every few weeks. By reducing WIP down to these small, two-week-sized buckets, we can greatly accelerate the delivery of high-priority features. This has been a fantastic revelation for traditional software development and has resulted in enormous improvements in delivery time, quality, and customer satisfaction. But as I explained previously, it may not yet be agile enough for help desk operations. Enter Kanban.

Kanban is a further step toward even smaller batch sizes and a move toward a more continuous flow model. Kanban eliminates the whole idea of iterations or sprints. In Kanban, like Scrum, we work on the highest-priority items, but when an item is done, we can deliver it immediately, sometimes within a day or even hours — very small buckets indeed (see Figure 1)! This seems like a perfect fit for help desk and support work. Requests come in, we actively prioritize them, we focus on the highest-priority items, and we deliver fixes often within hours. Through continuous reprioritization and continuous delivery, we can create a highly responsive organization without the headaches and overhead of trying to fit the work into Scrum's tidy little timeboxes.

Figure 1 — Kanban transfers work in even smaller batches than Scrum.

In Scrum, we achieve fast delivery through two-week deadlines of fixed scope. How do we ensure fast delivery in Kanban? We use WIP limits instead. By reducing WIP, we reduce cycle time (remember Little's Law). Admittedly, Scrum reduces WIP somewhat indirectly by using very short timeboxes; after all, you can fit only so much work within a two-week iteration. In Kanban, though, we eliminate the timebox entirely and instead use a simple, explicit WIP limit. We might say, for example, "The team will work on no more than five items at once, and we won't start another until we complete one of the five."

Remember that the more WIP we have, the longer it takes to deliver any particular piece of work. If we want very fast delivery, we could have a WIP limit of one and ask that the whole team focus on only one help desk ticket at a time. The result would probably be very fast delivery for this one item, but a lot of help desk tickets would go untouched, and overall throughput would probably suffer. If we wanted to address more help desk tickets simultaneously, we could have a WIP limit of 20. This would mean that the team could focus on the highest-priority 20 items at a time. In this scenario, 20 items would get some attention, but overall delivery time would suffer, since team members are juggling so many simultaneous tasks. The trick in Kanban is to discover a WIP limit — sometimes through trial and error — that strikes a balance between these two extremes. If the WIP limits are too high, then we have too much task switching, too much going on at once, large batch sizes, and slower delivery. If the WIP limits are too small, then we have very fast delivery but only for a very limited number of service items, and we probably have a lot of idle staff.

What about planning, communications, process improvement, and so forth? Unlike Scrum's prescription of daily stand-ups, retrospectives, point estimates, and the like, Kanban leaves a lot of these supporting structures fairly open, so it is even less prescriptive than Scrum from a process standpoint. If you want some additional structure and communications, you could borrow from both Scrum and Kanban as we did, or use additional tools, such as cumulative flow diagrams, kaizen workout sessions, and control charts.

By limiting our WIP and focusing on the highest-priority items, we were able to get much more work to an actual completed state.

OUR KANBAN IMPLEMENTATION

In the end, we decided upon a mixture of both Scrum and Kanban. We wanted the iterationless, continuous flow of Kanban, but we needed some additional support mechanisms to facilitate communications and continuous improvement. Our resulting process used:

- Kanban with a WIP limit of 14 (see below)
- Daily stand-ups
- Retrospectives
- Point estimates and burndown charts for the planned project work

Since DynamicMedia was already using Scrum for traditional software development, the idea of continuing to use daily stand-ups, retrospectives, and points was natural for the organization, and these practices were easily adopted by the Kanban help desk team. Our team had seven people on it, so we decided to try an initial WIP limit of 14. That way, each person would at most be working on two items at a time. This turned out to work pretty well; if one item was blocked for some reason, there was another item that each person could work on. The job of team leadership was to take all of the support requests each day and prioritize them clearly for the team.

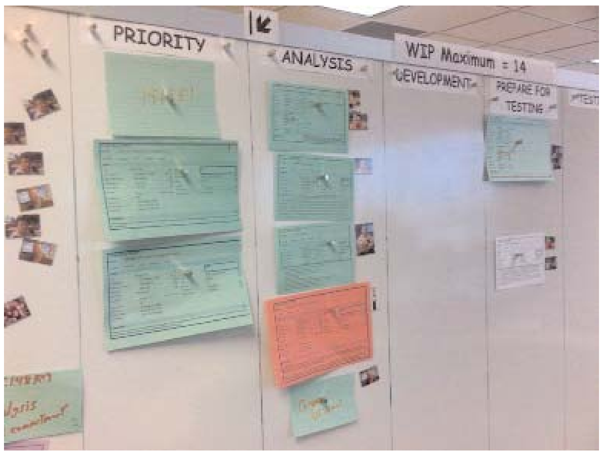


Figure 2 — We enforced the WIP limit through our planning board, which has several work states, such as analysis, development, and testing. The total WIP limit across all work states is 14.

Team members would meet each morning for a daily stand-up, review the priorities, and determine who was going to work on what. Periodically, we had retrospectives to see how the team was doing and where we could make improvements.

Realizations

To get started, we first set up a simple tracking board that had columns for Backlog, WIP, and Done (see Figure 2). We then asked the team to put all of the current work up on the wall so that we could visualize it. What we noticed, of course, was that there was a huge amount of work in WIP and only a few items in Done. Our goal was to turn that around. We instituted the WIP limit of 14, and within a few days we had our WIP down to our target level and many more items in Done. By limiting our WIP and focusing on the highest-priority items, we were able to get much more work to an actual completed state. And since these were the highest-priority support items, we knew that we were delivering the work items that would have the most impact on the organization as a whole.

Working this way, we noticed that our team leaders needed to be much more proactive and involved in reviewing the work and assigning clear priorities. This became one of their primary functions. Daily stand-ups have resulted in better visibility and cross-team communication, which the team has found valuable.

For actual planned projects, such as e-mail upgrades or hardware installs, we used typical Scrum point estimates and release burndown charts. Tasks related to these projects were intermingled with the support tasks so that if there were no high-priority support tasks, the team could focus on project work. After a few weeks, though, our visual system made it apparent that the project work was getting short-changed. While we had a lot of unplanned work in the Done column, and we were delivering outstanding support to our user community, our project burndowns showed that we were behind schedule for our planned projects. Clearly, user support work was taking priority over more strategic projects. While we knew that this was probably the case, the board combined with the burndowns highlighted the problem clearly and showed quantitatively how far we were behind on project work.

Kanban is a very simple process to understand — deceptively simple, in fact. But as is often the case, actual implementation was not a trivial undertaking. Initially, there was some confusion regarding what counted toward WIP and what did not, if and how we should deal with pair programming, how to handle items that were blocked, and so on. Kanban is more a concept or way of thinking than a "process." When we think of processes, we often think of highly detailed process steps, decision points, defined inputs, outputs, and the like. Kanban has none of these. It defines a high-level goal of moving toward single-piece flow and suggests some tools for achieving continuous flow, such as WIP limits. Beyond that, there is not a lot of implementation detail. Each team will need to evolve solutions for the kinds of issues mentioned above. Most importantly, each team will need to be patient as these issues are uncovered and addressed.

Next Steps

With the basic Kanban solution in place, and the realization that the strategic planned project work was not sufficiently addressed, the next step was to hold a team retrospective and then a short kaizen session to identify potential solutions and process improvements. The team came up with several ideas to help alleviate some of the problems. For example, one idea was to institute swim lanes for the two kinds of work — planned project work and unplanned support work. We could keep the WIP limit of 14, since that is working well, but divide it up into two parts: 10 for unplanned support work and 4 for the planned project work. This means that we would always have several planned project tasks being executed at any particular time, allowing us to start to make headway on the planned project work. By experimenting with these two WIP limits, we could find a balance between delivering outstanding service and delivering on the longer-term strategic projects.

Another idea was to develop a set of delivery metrics. Let's say that a help desk team has four types of work that they commonly deliver: large new features, small new features, large maintenance activities, and, finally, small maintenance activities. As new requests arrive on an almost daily basis, which is common in many organizations, the team would do a quick assessment and sizing of each. With some simple metrics, we could soon track the cycle times of the four kinds of work, so that the team would know that, on average:

- It takes 15 days (± 3 days) to deliver a large new feature.
- It takes 5 days (± 2 days) to deliver a small new feature.
- It takes 10 days (± 2 days) to deliver a larger maintenance request.
- It takes 3 days (±1 day) to deliver a small maintenance request.

Through these simple metrics and a continuous planning system, the team could make pretty accurate commitments and never have to stop the production line for iteration planning. With continuous flow, we do not have to figure out exactly how much work will fit within an iteration because there are no iterations — just a continuous flow of delivery to customers. But we need to be mindful that these delivery metrics would be valid *under our current WIP limit only*! If we change our WIP limit, then that will change our batch size, which in turn will affect the cycle time, and thus we will need to recalibrate our delivery times.

Telling customers that you cannot work on their item right now due to your self-imposed WIP limit is not always an easy conversation to have.

KANBAN CHALLENGES

Explaining the system to customers, particularly the idea of WIP limits, can be a challenge. Telling customers that you cannot work on their item right now due to your self-imposed WIP limit is not always an easy conversation to have. Of course, you can also tell them that once you *do* start on their problem, the WIP limits will protect them; you will be able to focus on their problem and get it done quickly! In my experience, customers like having some sort of well-working and simple system in place rather than the somewhat ad hoc approaches that seem to arise in the absence of a working solution.

Constant and clear reprioritization is a second major challenge. This is the real work in a Kanban system. Kanban forces management to make the tough calls about what truly is the highest-value work and what can wait. In large batch systems, these decisions do not actually need to be made. We can call everything "high priority" and work on it all at once in a long delivery cycle. Big batch systems allow us to say "yes" to everyone, but of course they all end up paying in longer-than-necessary delivery times. In Kanban, we need to be very clear about priorities, and this forces some very difficult discussions on an almost daily basis. However, the positive results of these hard decisions are clear — the truly highest-priority changes can be delivered to the firm with amazing speed and efficiency.

ONGOING EVOLUTION

DynamicMedia's Kanban system is in a continuous state of evolution. Through retrospectives borrowed from Scrum, the team sees issues that need to be addressed, such as:

- Changes to the prioritization scheme
- Additional states that need to be tracked
- Redefinition of what is in WIP versus what is in some sort of holding state (waiting, blocked, ready for deployment, etc.)
- Changes in WIP limits
- Further division of work into additional classifications/swim lanes

Process improvement is never finished, and Kanban is no exception to this rule. Any team that embarks upon this process transformation should plan on revisiting the process often, especially during the initial months.

BENEFITS

Team members and management at DynamicMedia were very satisfied with the Kanban approach and continue to use the system. The help desk team reports that the system provides an improved way of managing support work, better visibility, and better clarity on priorities and WIP. Kanban is a simple but powerful system for visualizing and managing work and is especially adept at managing unplanned work, a problem that has evaded solution for quite some time! WIP limits force prioritization and accelerate delivery with minimal process overhead and a high degree of responsiveness. The team has also reported that the Kanban system just "feels right." It is a very natural way for a support team to work that offers sufficient structure and visibility for managing the ever-changing priorities that are the essence of the help desk.

ENDNOTES

[1]Requirements are treated more independently in agile methods, with each going through the lifecycle somewhat independently of the others.

[2]See http://en.wikipedia.org/wiki/Little%27s_law.

Roland Cuellar is a leader in helping enterprise-level clients adopt the use of both agile and lean methods in their organizations. Mr. Cuellar has led numerous agile product development teams in a variety of areas, such as marketing applications, mortgages, financial compliance, media, and others. He has prior experience leading software development projects for Freddie Mac, IBM, Lockheed Martin, DHL, CC Pace, and LitheSpeed LLC. Mr. Cuellar is Director of Program Management for Three Pillar Global in Fairfax, Virginia, USA. He can be reached at Roland_Cuellar@yahoo.com.

Use of Kanban in Distributed Offshore Environments

by Siddharta Govindaraj and Sreekanth Tadipatri

Over the last few years, agile processes have gone from being used primarily by single-location colocated teams to being adopted by multilocation distributed teams. These processes have enabled businesses to respond faster to rapidly changing market conditions. At the same time, the market for IT and business process offshoring — which had already been sizable — has grown exponentially. While combining agile with offshoring can allow companies both flexibility and a global reach in terms of services work, it can also present many challenges. In this article, we explore how Kanban can be used as an alternative to first-generation agile methods in offshore environments. We also discuss Kanban anti-patterns that we have seen in such environments.

CONFRONTING THE CHALLENGES OF OUTSOURCING

Management consulting firm A.T. Kearney studied more than 40 companies that had tried offshoring and concluded that there were significant benefits to be achieved from it, though the overall success rates varied. The parameters studied were capacity, organizational flexibility, revenue performance, service levels, process maturity, and organizational capability.[1] While companies have had mixed results,[2] it is clear that outsourcing is here to stay.

Some of the reasons cited for difficulties in outsourcing are:

- Lack of visibility and transparency into the status of the project, which leads to a mood of mistrust between the customer and the vendor
- Differences in business culture between onsite and offshore locations
- Time zone differences between locations
- Inflexibility in adapting to changing requirements

Agile processes provide solutions to some of these problems. They advocate collaboration over contract negotiation to encourage increased trust. They recommend frequent incremental delivery of working software, which provides some visibility and keeps everyone up to date on where the project actually stands. This in turn enables stakeholders to make changes to requirements midway through the project.

However, offshoring combined with agile comes with a few challenges of its own:

- The close-knit, cross-functional teams required by agile methods are difficult to implement in a distributed environment.
- Timeboxed development is not suitable for many types of offshore engagements, such as maintenance and support or projects where only some activities (e.g., testing) have been outsourced.
- Large-scale organizational changes are required for agile implementations to succeed. When these changes don't happen, it leads to a dysfunctional organization.
- Agile adoption can magnify existing issues with time zone gaps and business culture differences.

The question is, how can executives merge the feedback and responsiveness of agile with the benefits of outsourcing? We suggest that Kanban can be used as a method for incremental improvement that can work with, arguably, any underlying process. It can be effectively applied in a variety of contexts, including new development, maintenance and support, and environments in which there are handoffs or blocking dependencies between teams. Perhaps more important, Kanban can be applied without the need for transformational change, which makes it possible to use Kanban principles on any existing project.

APPLYING KANBAN TO OUTSOURCED PROJECTS

Kanban for software development was originally conceived as a solution to the difficulties in managing distributed offshore teams.[3, 4] In this section, we will look at strategies for improving outsourced projects through the application of Kanban steps.

Visualize the Workflow

Just putting all the steps in the workflow onto a Kanban board and visualizing the state of each work item can lead to many positive changes. Transparency is increased, which leads to greater trust between onsite and offshore teams. Furthermore, this simple visualization can help you identify bottlenecks in the workflow. You might be surprised where bottlenecks turn up.

Experience Report 1

In one project that Siddharta worked on, the bottleneck was actually outside the development team. Many stories were awaiting user acceptance testing from the customer.

We were initially unaware of this bottleneck, but once we started tracking the number of work items in each state across the whole value stream, the problem became visible. We were then able to work with the customer to speed up the acceptance process.

A quick way to reduce the lead time is to see if there are queues that can be reduced or eliminated.

Experience Report 2

Before visualizing work items on the Kanban board, team members were unaware of what others were working on. They would often start a new work item instead of completing one already in progress. This meant that a lot of new work items would get started.

After the offshore site set up a Kanban board for local team members, they would scan the board to see whether they could help complete an existing work item before pulling a new one. Visualizing the workflow enabled team members to self-organize in this way.

Later, the project sponsor instituted an electronic Kanban board for the entire project so that teams across locations could use this single board to reflect all their work. This enabled tacit communication between team members in all locations as well as with the business and sponsor — everyone on the project had visibility into any work item (i.e., the item's state of completion and the team that was working on it). Thus, communication challenges and time spent coordinating work were minimized.

Look at Queue Time

In any software development project, regardless of the methodology, items often spend a lot of time in queues waiting to be worked on. This queue time can be a substantial part of the overall lead time. (Lead time is the amount of time that elapses from the moment a task enters the work system to the moment it is delivered to the customer.) It is common for a work item to spend more time waiting in queues than actually being worked on.

Points of handoff tend to be prime candidates for long queues. A quick way to reduce the lead time is to see whether there are queues that can be reduced or eliminated. If you can't eliminate the handoff, then setting limits on queue length can also help, even if it means stopping upstream activity until capacity is available downstream.

Experience Report 3

In one project, every piece of code in certain key components had to be reviewed by someone else. Often, a feature would be implemented in a few days and then wait for almost a week until someone else got around to reviewing it. As a result, the total time to implement a feature was almost two weeks, of which only two or three days were spent in actual implementation. The rest of the time was spent waiting for review.

We implemented pair programming for code changes that involved these components. This eliminated the need for code review, and we were able to deliver features almost a week earlier.

Experience Report 4

One highly successful distributed agile implementation that Sreekanth experienced involved eight teams of eight members each. The teams in the offshore location were working in silos even though they were on the same project.

Just bringing two teams at the offshore location into one room and putting up the work items in slots to reflect their current state of readiness helped the team tremendously. It soon became clear that when the developers finished coding the stories, the testers could not keep pace with testing — neither exploratory testing nor automation of functional and acceptance testing. Thus, a lot of items got queued up in the testing phase. When the developers and business analysts saw that testing was a bottleneck, they pitched in by doing manual exploratory testing and automating functional and acceptance test cases. Team maturity improved in terms of self-organization and collaboration, resulting in each work item completed at the earliest possible time.

Make Policies Explicit

Most organizations have implicit policies about how different types of work are to be handled. Kanban teams make these policies explicit so that they are applied consistently.

Experience Report 5

In one maintenance project, due dates for tickets were defined according to service-level agreement (SLA) terms. Originally, when deciding who should work on which ticket, management would assign similar work items to particular individuals. This resulted in an uneven distribution of work, where certain team members would have to multitask on many tickets while others were relatively free.

After adopting Kanban, the team decided on a policy that whenever someone became free, he or she should pick the ticket that is closest to the SLA due date. With this policy in place, team members no longer needed someone to assign tickets to them. When they finished a task, they simply picked the next ticket in the queue. This explicit policy allowed the team to self-organize and manage the flow of tickets themselves.

Limit Work in Progress

In order to achieve optimal lead time, we must limit the amount of work in progress (WIP).[5] Limiting WIP reduces the amount of multitasking, resulting in shorter lead times.

Experience Report 6

After setting up the Kanban board in one organization, it became clear to the director of operations that team members were constantly overcommitted, often multi-tasking between many work items. This had the effect of increasing lead times on the work items, causing missed due dates.

In response, the team, in consultation with the director of operations, established WIP limits. Setting up WIP limits eased the pressure on the team, since the stake-holders could not overload them when limits were full. WIP limits also encouraged team members to complete stories in progress instead of picking up new stories, resulting in decreased lead times.

OFFSHORE KANBAN: PITFALLS TO AVOID

So far, we have talked about successful applications of Kanban. That's not to say that there are no pitfalls. Here are some things to look out for:

- **Not adopting a WIP limit.** Limiting the work in progress is a fundamental feature of Kanban. However, this is where the maximum resistance is seen. Managers often have the perception that limiting WIP leads to reduced capacity utilization. Executives may also be tempted to compare WIP limits across locations.
- **Maintaining static WIP limits.** WIP limits are not supposed to be fixed in stone. They can and should change, taking into account factors like changes in team composition.
- **Not managing the flow of work.** If the flow of work is not managed, teams could revert back to the earlier behavior of taking on new work instead of finishing work in progress.
- **Not removing impediments.** Simply visualizing the work on a Kanban board does not yield reduced lead times. Teams must be ready to regularly take action and make improvements based on findings from the board.
- **Not updating the Kanban board regularly.** If the Kanban board is not updated, it may not reflect the current state of work items.
- **Having no available Kanban expertise.** Kanban is relatively new. It can be difficult to find people and teams that are proficient with it. At the same time, Kanban is less prescriptive than other agile processes such as Scrum. This can make it more difficult to get buy-in from management.
- **Not mapping the value stream completely.** Sometimes the value stream is not mapped from concept to delivery. This leaves the risk that dependencies outside of the team will not be taken into account.
- **Having overly specialized teams.** Teams may find themselves getting the same type of work based on their specialization. For example, one team develops the system while another team does the testing. The lead time to get the required functionality to the customer is increased, since there are multiple handoffs and queue times. While this might be the starting position, less specialized teams can lead to reduced lead times.

- **Not having sufficient stakeholder involvement.** This is perhaps one of the most critical factors in determining the success or failure of Kanban adoption. Oftentimes when a team adopts Kanban, the stakeholders have no interest in knowing how it works or benefits the organization. In this situation, change never occurs.
- **Creating silos of knowledge.** Project managers in matrixed organizations often route work to teams that deliver the projects they manage. The knowledge of the system gets siloed among teams. No longer can any team work on any story. Instead, it is possible that one team may be overloaded while another is idle. Also, team members could lose motivation when repeatedly working on similar stories.

One of the principles of the Kanban method is to continuously improve, and to do this, teams need to believe that there are *always* better ways of doing things.

CULTURAL CHALLENGES TO KANBAN

As with any new initiative, culture plays an important part in how successful a Kanban implementation will be. Here we consider some cultural factors that pose a challenge to Kanban adoption.

Micromanagement

In some cultures, there is a reluctance to deliver the bad news early, but this information is important to all the stakeholders to mitigate business risks. Kanban makes the status of work visible. Problem areas and bottlenecks are clearly apparent, and some team members may fear that they are being micromanaged through the transparency.

Inflexible Belief in Best Practices

At some workplaces, there is an inherent need to define "best practices" and a lack of willingness to identify problem areas and make needed improvements on a continuous basis. Therefore, when a team either opts to use Kanban or is required to, it is likely that the initial workflow mapped on the board will be taken as is and that no changes to the workflow will be made based on experience and the knowledge gained by the team members.

One of the principles of the Kanban method is to continuously improve, and to do this, teams need to believe that there are *always* better ways of doing things.

Lack of Team Empowerment

Team members in some organizations are not empowered to alter the Kanban board or WIP limits, and they look to their manager to suggest any improvement. In one case, the team pitched in with the estimates but then asked their manager, "Which task do you want us to start working on?"

In matrixed organizations, project management often controls the work done by the team and assigns work to team members. Consequently, team members are not motivated to learn better ways of working, and Kanban becomes meaningless.

Fear of Perceived Incompetence

In the software industry, especially in the services sector, there may be a tendency to "protect" the junior developers from being "exposed" to the customer. One junior developer admitted that he was afraid of being perceived as incompetent, since he took much longer to complete a task than a senior member of the team. Kanban makes these differences visible and thus may cause an individual to feel insecure.

The Politics of Change

As with any change effort, adopting Kanban can be a political minefield in an organization. Any change being brought about will be embraced with enthusiasm by some and fought bitterly by others, depending on the personal equation of the people involved — both at the leadership level and at the middle management and team levels. Success or failure depends a lot on who in the organization supports the effort. Each geographical location in a company may try to establish an identity for itself, and adoption of Kanban at each location may depend on how it fits with the location's identity.

CONCLUSIONS

When done properly, Kanban creates an environment with the following characteristics:[6]

- There is visibility into the development process.
- Team members are actively involved in controlling their tasks and participating in collaborative problem solving.

- There is a culture of continuous improvement.
- There is increased trust between customers and vendors.

Through a few simple steps, Kanban provides teams with a mechanism to continually improve their process, cut lead times, and improve responsiveness to rapid changes in the market. The key ingredients for the success of offshore Kanban are stakeholder involvement; an empowered, cross-functional team; and involvement of the whole value chain.

ACKNOWLEDGMENTS

The authors would like to thank Hedwig Baars, Hitesh Ramnani, and Govindarajan Sundararajan for sharing their experiences with offshore Kanban for this article.

ENDNOTES

[1]"What to Move Offshore: Selecting IT Activities for Offshore Locations." A.T. Kearney, 2004 (www.atkearney.com/images/global/pdf/What_to_Move_offshore_s.pdf).

[2]Love, Jim, and Darrel Berry. "Backsourcing vs. the Hotel California Syndrome." Cutter Consortium Sourcing & Vendor Relationships *E-Mail Advisor*, 26 January 2011.

[3]Anderson, David J. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

[4]Anderson, David J., and Dragos Dumitriu. "From Worst to Best in 9 Months: Implementing a Drum-Buffer-Rope Solution in Microsoft's IT Department." Paper presented to the *TOC ICO World Conference*, Barcelona, Spain, November 2005.

[5]Maeda, Masa K. "WIP: From Limiting Software Development to Balancing the Project." Cutter Consortium Agile Product & Project Management *Executive Update*, Vol. 11, No. 4, 2010.

[6]Stevens, Dennis. "Kanban, Mental Models, and Double Loop Learning." *Dennis Stevens: Enabling the Agile Enterprise*, 14 July 2010 (www.dennisstevens.com/2010/07/14/Kanban-mental-models-and-double-loop-learning).

Siddharta Govindaraj is the founder of Silver Stripe Software Pvt. Ltd., a company that develops products for teams that follow agile and Kanban. Based in Chennai, India, Mr. Govindaraj has six years of experience with agile, of which the last three years have been in applying Kanban. He was a speaker at the Lean Software & Systems Conference 2010 in Atlanta. Mr. Govindaraj can be reached at siddharta@silverstripesoftware.com.

Sreekanth Tadipatri has been a practitioner of software development, software quality assurance, and project management for more than 12 years. Based in Bangalore, India, Mr. Tadipatri has helped more than 50 projects and 100 distributed teams transform themselves by adopting agile. During his coaching engagements, he has experimented with introducing Kanban to agile teams. Mr. Tadipatri can be reached at sreekantht@agilefaqs.com.