NetObjectives
ESSENTIAL WHITE PAPERS

# Demystifying Kanban

by Al Shalloway

**Demystifying Kanban**

**A Net Objectives Essential White Paper**

Net Objectives Press, a division of Net Objectives

1037 NE 65th Street Suite #362

Seattle, WA 98115-6655

404-593-8375

Find us on the Web at: *www.netobjectives.com*

To report errors, please send a note to *info@netobjectives.com*



Cover design by Andrea Bain

Published by Net Objectives, Inc.



10 9 8 7 6 5 4 3 2

Printed in the United States of America

**This article was first printed in Cutter IT Journal, March 2011, Vol. 24, No. 3, pp. 12-17.**

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 different 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, time-boxed 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 time-box.

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

**Alan Shalloway** is the founder and CEO of Net Objectives. With over 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*.

team level and, more importantly, 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 time-boxing, 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.

Time-boxing 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 customer support.
4. Create a board that reflects the team's flow of work.
5. Educate the team on the principles of flow and pull.

#### 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.

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:

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

**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 under- stands 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 things 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 the policies.

**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 does not require time-boxing 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 time-box.

## 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 [at the wide scale].

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.

## 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 off-load 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 continuously. 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 continuously. 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]

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

## 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

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.

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 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.

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, time-boxed 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 to 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 the team 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.

## 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 being 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 are those who found other Agile methods difficult to apply. The second are those who were late adopters to 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 sizeable 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 they:

- Make all of their work visible
- Limit their work in progress to within 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 Succeeds." Net Objectives, 24 May 2007 (*www.netobjectives.com/blogs/challenging-why-not-if-scrum-works*).
2. Shalloway, Alan. "How Successful Pilots Actually Often Hurt an Organization." Net Objectives, 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 on the differences between Kanban and Scrum, see: Shalloway, Alan. "The Real Differences between Kanban and Scrum." Net Objectives, 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 entitled

# Drive from Business Value

## BUSINESS-DRIVEN SOFTWARE DEVELOPMENT

**Business-Driven Software Development** is Net Objectives' proprietary integration of Lean-Thinking with Agile methods across the business, management and development teams to maximize the value delivered from a software development organization. This approach has a consistent track record of delivering higher quality products faster and with lower cost than other methods.

Business-Driven Software Development goes beyond the first generation of Agile methods such as Scrum and XP by viewing the entire value stream of development. Lean-Thinking enables product portfolio management, release planning and critical metrics to create a top-down vision while still promoting a bottom-up implementation.

Our approach integrates business, management and teams. Popular Agile methods, such as Scrum, tend to isolate teams from the business side and seem to have forgotten management's role altogether. These are critical aspects of all successful organizations. Here are some key elements:

- Business provides the vision and direction; properly selecting, sizing and prioritizing those products and enhancements that will maximize your investment.
- Teams self-organize and do the work; consistently delivering value quickly while reducing the risk of developing what is not needed.
- Management bridges the two; providing the right environment for successful development by creating an organizational structure that removes impediments to the production of value. This increases productivity, lowers cost and improves quality.

## BECOME A LEAN-AGILE ENTERPRISE

**Involve all levels.** All levels of your organization will experience impacts and require change management. We help prepare executive, mid-management and the front-line with the competencies required to successfully change the culture to a Lean-Agile enterprise.

**Prioritization is only half the problem**. Learn how to both prioritize and size your initiatives to enable your teams to implement them quickly.

**Learn to come from business need not just system capability**. There is a disconnect between the business side and development side in many organizations. Learn how BDSD can bridge this gap by providing the practices for managing the flow of work.

## WHY NET OBJECTIVES

While many organizations are having success with Agile methods, many more are not. Much of this is due to organizations either starting in the wrong place, such as focusing on the team when that is not the main problem, or using the wrong method, such as using Scrum or kanban because they are popular.

Net Objectives is experienced in all of the Agile team methods (Scrum, XP, Kanban) and integrates business, management and teams. This lets us help you select the right method for you.

## LEARN TO DRIVE DEVELOPMENT FROM THE DELIVERY OF BUSINESS VALUE

What really matters to any organization? The delivery of value to customers. Most development organizations, both large and small, are not organized to optimize the delivery of value. By focusing the system within which your people are working and by aligning your people by giving them clear visibility into the value they are creating, any development organization can deliver far more value, lower friction, and do it with fewer acts of self-destructive heroism on the part of the teams.

## THE NET OBJECTIVES TRANSFORMATION MODEL

Our approach is to start where you are and then set out a roadmap to get you to where you want to be, with concrete actionable steps to make immediate progress at a rate your people and organization can absorb. We do this by guiding executive leadership, middle management, and the teams at the working surface. The coordination of all three is required to make change that will stick.

## OUR EXPERTS

Net Objectives' consultants are actually a team. Some are well known thought leaders. Most of them are authors. All of them are contributors to our approach.


*Al Shalloway*


*Alan Chedalawada*


*Guy Beaver*


*Scott Bain*


*Max Guernsey*


*Luniel de Beer*

## SELECTED COURSES

### Executive Leadership and Management

Lean-Agile Executive Briefing

Preparing Leadership for a Lean-Agile/SAFe Transformation

### Product Manager & Product Owner

Lean-Agile Product Roadmaps

PM/PO Essentials

### Lean-Agile at the Team

Acceptance Test-Driven Development

Implementing Team Agility

Team Agility Coaching Certification

Lean-Agile Story Writing with Tests

### Technical Agility

Advanced Software Design

Design Patterns Lab

Effective Object-Oriented Analysis and Design

Emergent Design

Sustainable Test-Driven Development

### DevOps

DevOps for Leaders and Managers

DevOps Roadmap Overview

### SAFe®-Related

Implementing SAFe with SPC4 Certification

Leading SAFe® 4.0

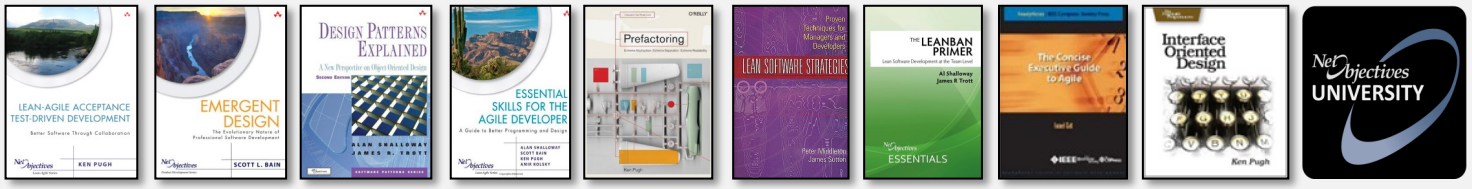
Using ATDD/BDD in the Agile Release Train (workshop)

Architecting in a SAFe Environment

Implement the Built-in Quality of SAFe

Taking Agile at Scale to the Next Level

## OUR BOOKS AND RESOURCES





CONTACT US
info@netobjectives.com
1.888.LEAN-244 (1.888.532.6244)


LEARN MORE
www.NetObjectives.com
portal.NetObjectives.com

Net Objectives