Preliminary draft – Agile 2011 – 8 Feb 2011

Scrum and Lean : How a Lean Scrum Can Improve Your Performance

Jeff Sutherland & Hugo Heitz

Abstract

Scrum's innovative way of transforming the way we work is a disciplined approach to smoothing out flow, eliminating pressure, and waste removal. The co-creator of Scrum and a lean expert will examine Scrum and show why violating lean principles will cripple a Scrum implementation. A practical case study will show how approaching Scrum from a lean point of view can double productivity and quality throughout a large organization. We show how lean metrics used to drive process improvement are easy to implement by beginning Scrum teams to optimize acceleration in performance.

Introduction

The origins of Scrum were deeply affected the work of Takeuchi and Nonaka [1] who observed that great teams at Honda, Fuji-Xerox, 3M, and HP exhibited behavior similar to Rugby teams, particularly the Scrum formation in Rugby. The two grandfathers of Scrum formulated the concept of a knowledge creating company [2] and described how a product emerges from knowledge generated by the dynamic interaction of a team. Nonaka's analyses of the Toyota Production System [3] shows how knowledge creation works in a lean environment. While Scrum is based on complex adaptive systems theory, the self-organization that Nonaka observed in high performing teams was seen in lean companies. Scrum is indirectly affected by lean, yet virtually everything in Scrum can be explained by lean concepts. Scrum implementations can be improved dramatically by those who understand lean.

Lean is the name given by Jim Womack and Dan Jones to the Toyota Production System in their book "The Machine That Changed the World." Thus Lean is the western view of what is going on at Toyota and can be seen as a lot of things : a business strategy, a methodology, a set of tools, a set of practices. Lean is a way to deliver to customer exactly what he wants, when he wants, where he wants just in time. Lean is a way to reduce costs by solving problems and especially exposing those practices that create waste. Lean is a way to empower people, to develop them to their full potential by solving more and more complex problems. Scrum is a way to implement all these things and many lean experts view Scrum as the best implementation tool for introducing lean practices throughout a company.

Scrum and Lean converge on numerous aspects and it is enlightening to see what Scrum shares with the Lean approach. Each can inform the other to enable better implementations of both Lean and Scrum. This paper an will provide a lean analysis of Scrum to show why each piece of Scrum works from a lean point of view. It will also clarify how compromising lean principles will cripple your Scrum implementation.

The two pillars of lean : Respect for people & Continuous improvement

Womack and Jones based their concept of lean on the Toyota Production System (TPS) in the 1990's. Invented by Taiichi Ohno, the two core concepts of TPS were JIT and jidoka. Jidoka relates to equipment intelligent enough to shut itself down. JIT is a system with little inventory where people shut down their line when there is a problem. Thus lean is often looked at as systems of tools and techniques. However, Fujio Cho, the president of Toyota observed that competitive advantage came from engaging all the people of a company in continuous improvement. When he published The Toyota Way 2001 he changed the two pillars of lean to **respect for people** and **continuous improvement**.

1.1 Respect for people

Respect for people means Teamwork and Human consideration. As explained in the Agile Manifesto, the Scrum team finds one of it's engines in **Individuals and Interactions.** It is people who create value, not processes.

At Toyota, there is a saying "Life goes by second after second and every second is important." Scrum values time as it focuses on features that are the most valuable to users, estimated by the people who do the job, delivering software that works during each sprint.

From the lean perspective, software built in an iterative, incremental way is like building in small batches focused on customer value. The "sprints" in scrum are relatively short amount of time (1 to 4 weeks) in which the team develops software that works.

What you get from a lean point of view is opportunities to learn and to get a better understanding of the customer needs. Because the customer does not know really what he wants unless he can see it in a demo, Scrum adopted a mantra from the MIT Media Lab: "Demo or Die!" The learning is a Sprint Review is sharing between developers and customers.

There are many ways that you show respect for people in Scrum. As opposed to the waterfall model, there is no GANTT chart planning listing everything that needs to be done. In Scrum, the best way is to get customers in the room and if you can't then find a product owner to represent the customers needs.

The product owner will be your proxy to the customer needs and you will try to develop something that will run at the end of a release, and you will strive to get releases as short as possible. This is respect for human life.

For the customer, it is better to see as fast as possible that what is developed is not at all what he had in his mind. Words are a weak way to communicate, the best is to try it in real work. For the developer, it is better to develop something that has value for the customer and to confirm this as fast as possible before wasting more time if things go wrong. For the shareholder, it is better to know as fast as possible that the market is there or not before wasting cash to create vaporware.

Scrum can be seen as a way to avoid waste in many ways. As lean is a way to improve in a continuous way, especially working on waste (which reveals problems), Scrum and Lean point in the same direction.

Teamwork
The scrum practice is built to run a team. A team is a collection of people trying to achieve a common goal.

Contrary to the traditional command-control model which works for mechanical and simple operations, you need another type of environment if you want people to commit to a goal, achieve objectives and be fully engaged. As explained by Dan Pink in "Drive", you need to have a fair compensation (difficult to create and innovate when you don't know if you can afford a roof other your head and decent food), a goal that is higher than only revenue and a team environment. The team, the collection of people you work with every day needs to be why you come to the job with good feelings.

A lean team is viewed as a collection of 3 to 7 people. There are numerous questions about the optimum team size, a good way to answer it is "how many people can you have on a table to get a great conversation ?"

The lean team is led by a team manager who is not the manager as a command and control authority, but a mentor, a coach who will challenge his team and every single one person to develop his potential to his maximum. People need to work towards mastery, to be the best they can be.

In Scrum, this role is assumed by the ScrumMaster. The scrum master role is not to monitor or report but to prepare the team for peak performance. To achieve that, the ScrumMaster's role is to help the team. In fact it reverts the command-control traditional hierarchical pyramid because the people who create value (the developers) are in this approach at the top and the management staff is there to help.

The scrum master for example must protect the team from interruptions. What does this mean from a lean perspective?

First is alignment, or *hoshin kanri* (strategy deployment). Once you set up the sprint, you won't work on other issues and will focus on what are the right things to do to achieve business objectives. By getting agreement on the scrum approach before you start to run, you engage management to release pressure on things that are not top priority.

You are putting quality and delivery on the table and explaining to the people who wants to ask you something else why they can't have reports, extra demo or other things that can be done but that are not the right things to do.

Everyone is doing things in our companies, but are we getting the right things done ? This is essentially what lean explains in lean management.

The problem solving approach creates a pull-based authority in both Scrum and lean to pull the resources of the company and transform the ways of doing things to better ones.

Avoiding two specific type of waste

In the occident we focus on a specific type of waste which is muda, because it is visible, you can see it, you can touch it. But to create muda you need other types of wastes that are more subtle and insidious.

Mura – unevenness and Muri - overburden

To create a great performing & learning environment, you need an environment where you will be able to learn. You can't ask a firefighter to learn how to handle a fire if you take him out of high school and get him in the first fire you see. This is more or less what companies do everyday with their own people, they push them to the tasks and see if they survive.

If you want to show real respect to your people, it is not only in words like "our people are the most valuable asset of the company" but also in reality by creating an environment where people are expected to develop their abilities to their full potential.

That means that you need to be pretty sure that day to day you will do the same thing in order to be able to understand it better and to improve. If the flow production is not smoothed out and leveled, you will run from crisis to crisis, from bug correction to new feature.

How to hide mura

Organizations try to hide mura by creating extra inventory. When you create more inventory, you create muda. To remove muda, you need to work on mura.

The most crippling example is creating software that is not tested. Then you have an inventory of unfinished work. Finishing the work at a later day always costs at least twice as much testing and bug fixing as completing it immediately and may cost up to 24 times as much work. Because testing is usually the constraint on delivering, deployment may be delayed from 2-24 times.

In the ScrumMaster environment, how do you work on mura?

You don't assume the speed (ie the capacity) of the team to deliver. You discover it in the first sprint. Let's assume you have a team that delivers 20 points. Not 30, not 40. The reality is 20 points. You need to face it and assume that for now you can deliver 20 points at each sprint. Once you know your speed, you will level your production to be sure that you won't go too slow (no stretch zone, no challenge for the team) or that you won't go too fast (overcapacity, burn-out, bugs, etc…)

Remember : the goal is to get teams who deliver, who learn, who will be able to improve and innovate – we will get back to it later. In lean, that means you will use leveling (or *heijunka*) to create a stable environment.

In Scrum, that means you will put in the backlog (the features list you will take in the next sprint) the features that are the most needed/valued by customers and corresponding to the capacity of

the team once you estimated each feature using planning poker and ensuring a buffer for “extra” or unknown work, which means you won’t assume you will work to your full capacity. By doing that you allow, you create space for people to think, to understand and to improve the way they do things.

In lean this also refers to a concept called *takt* time. The *takt* time is the tempo given by the customers to the production line once leveled to produce each item. For example, if you need to produce 60 cars in a 6 hours, you will try to produce 10 cars per hour or 1 car every 6 minutes. Tic tac tic tac, the customer leveled demand gives the tempo to production.This works for production because it is product based, but for IT projects how can you translate that?

Within projects you need to produce according to customer needs (this is the job of the “backlog” which prioritizes the features) at a certain pace (this is the job of the sprint approach which allows to confirm the understanding and accelerates the learning) according to the team’s capacity (this is the points capacity of the team).

Within a release (i.e., a certain number of sprints) you need to deliver a product to the customer, you will know pretty fast if your team is able to deliver it or not (after 2 or 3 sprints you will know better) and therefore you can adapt.

There are multiple ways to adapt. Let’s assume that you understand really fast that you won’t be able to deliver at the rhythm you are able to deliver. One (the traditional way) is to get help, in fact get more people to do the job.

The other is to work on impediments to improve team productivity and get more work done with the same people. The second approach is the core of lean and we will come back to it later.
For the moment, let’s assume that the amount of points needed for the release is compatible with the team’s capacity and the amount of time available. If the release plan looks good, how do you ensure that you are and keep right on track?

The burndown chart gives you the amount of points needed to be “done” by the end of the sprint. Another way of reading it is the amount of points needed to “done” at every milestone of the production.

You can read : we, as a team, need to produce 60 points in 6 days, or we, as a team, need to produce 10 points every day.

So *takt* time gives you the amount of points needed to be “done” for every period of work.
With the scrum daily meeting, you follow your production every day and ensure that the team knows if you are ahead, on track or late about the work and most important, act on this information.

With the board, you allow features to be pulled rather than pushed, this is a *kanban* board working as within a supermarket, you enter new features once the others have been completed.

Scrum provides practices that allows the team to avoid mura and muri, these practices helps the team to focus on important aspects for the customer, to pull the features according to the rhythm

of the customer, prevents the team to work on non value added demand, these practices also include other aspects that are more subtle. A

"Don't ship junk"

One aspect of lean is the importance of building quality in the product, directly in the code, and to never allow defect to go further down the line to the customer. How can we make that happen?

The team needs to see as soon as possible that a defect is present, this is where the conception of "good" is important. What is a good job? What is a good code? If the team is able to understand it, then the team will be able to clarify the criteria to start to code in good conditions (definition of ready), to ensure that a task is fully completed (definition of done) and that there are no regression problems in the code that is produced.

What does it mean in lean ?

In order to avoid bug, you need to create a mistake proofing device, a test harness, which is called "*poka yoke*". The best is that the mistake proofing device prevents the mistake (for example, the USB cable on your laptop, try to connect the wrong cable and it won't be allowed), but you can also create an alert if the mistake proofing detects something wrong. This is the role of test driven development, you write the how to demo, you write the test code first, then the code. The interesting thing is that when you coach your team that way of thinking you are validating quality as the most important aspect of the job rather than productivity itself. It is easy to write lines of code that are worthless or bugged, it is harder to write good code that produces no bug. Of course this means that tests must be run in a continuous way, this is where continuous integration takes place. The continuous integration systems allows tests to be run automatically, warning if a bug occurs and preventing the team to accumulate more and more errors before reacting.

We have seen that scrum and other practices allows a team to learn faster and to provide better code. All of this is based on respect for people, not having them doing unnecessary work, not allowing them to be randomly disrupted, helping each other to have respect for the customer, and making the world a better place to live and work.

An interesting oversight that is common in westerners when looking at lean is "there is no *kaizen* without *hansei*. *Hansei* is a deep sense of regret/repentance/shame for dysfunctional behavior, channeled toward positive redirection. It is something that too often is lost in the translation from Japanese Lean into American Lean and through to Scrum." [4]

We will now enter what is the other pillar of lean, the one that creates a competitive difference - continuous improvement.

Continuous Improvement

The real difference with lean is that it tries to provide every single person the capacity to do "*kaizen*", the Japanese word for continuous improvement.

In fact, it redefines the job as: Job = work + kaizen

That means that every one has a responsibility not only to perform his task but also to search ways to improve how to perform his task. There are only good opportunities to work on problems because problems in the system create wastes.

Why is it so important to create a good environment?

It is difficult for people to recognize that there is a problem, not only in their company or in their team but most off all in their mind, because in the end it is the way we think that makes us act or work with certain behaviors. The important part of continuous improvement is to allow us to change our way of thinking in a controlled environment using a scientific approach.

There is no way to convince people by talking to them, the only efficient way is that they try something themselves and see the results. We learn by doing, we change over time because we repeatedly practice something, in the end it makes us change our mind and our way of thinking.

How do did you learn how to ride a bike ?

Not with a powerpoint presentation, you learn by trying to move ahead, to remove one foot from the ground, then two, you fall…and fall again…and in the end you know how to ride a bike.

Lean provides a scientific approach to understand a problem, tools to grasp the situation, techniques to fix some issues and get results. If you stay on tools and techniques, you will not get the results you want in the long term.

Lean is not about processes, tools or periodic reviews. Lean is all about people because it is people who think and are able to solve problems. Lean is about getting every people to do continuous improvement within their jobs.

How to get continuous improvement in the real world as an every day activity?

The only way to make it is to have a practice, something simple that you can apply anytime, in any part of the job - a "*kata*". What teaches the *kata*?

The *kata* is made of 5 questions :

1) What is the challenge ?
2) What is the actual situation ?
3) What are the obstacles preventing us to achieve the challenge ? What is the one I address now ?
4) What is your next step ? (beginning of a PDCA cycle [5])
5) When can we go and see what we learn from doing that step?

An interesting thing is that you can see that the *kata* is simple, positive, and also relies on the same approach than the one described by Dan Pink in "Drive" [6] where he explains what people need to have to perform in an intellectual environment : autonomy, mastery and a goal.

The goal = what is the challenge ?

Autonomy = we belong to a team where everyone will help to perform
Mastery = we understand and are able to deliver or are able to learn what we need to deliver

Let's talk about one thing that is really interesting in scrum in a lean way : the technical debt. As the code goes more and more bigger, the system goes more and more complex and the technical debt increases, you will notice that velocity goes down.

The first thing that lean teaches is "Stabilize first".

If I have to do comparisons, I would say :
In lean :
Job = work + kaizen
In Scrum :
Code = New features + continuous improvement on existing code and ways to create new one

To stabilize your technical debt, you will need to refactor your code, to put in continuous integration, maybe per programming, etc…

Once the practices that ensure the technical debt is stabilized, you will get steady velocity.
In fact, at this point you are already doing continuous improvement because you are creating new features without increasing the technical debt, that means you are deleting (getting old stuff out), simplifying (better algorithms) your existing codebase.

After a certain amount of time, there will be a shift within your mind, you will get that continuous improvement is the key to the future. Continous improvement is not optional. It not only guarantees the way to succeed but also guarantees survival.

As the team begins to improve, your team will enter a zone where challenges need to be overcome, it is the role of a great scrum master to create those challenges, to awake kaizen spirit in every single member of the team, to solve problems that are more and more complex. By solving problems, your team will increase its capacities, increase its velocity.

The interesting part of lean is that it is a practice that is able to renew any single approach that is already available, giving us tremendous capacities to learn and to adapt.

The challenge for scrum is huge if scrum wants to integrate lean thinking totally - is Scrum able to **Scrum the Scrum**? Can we use Scrum to improve Scrum.

The future will tell us…

References

[1] H. Takeuchi and I. Nonaka, "The New New Product Development Game," *Harvard Business Review,* 1986.
[2] I. Nonaka and H. Takeuchi, *The Knowledge-Creating Company : How Japanese companies create the dynamics of innovation:* . New York: Oxford University Press, 1995.
[3] H. Takeuchi and I. Nonaka, *Hitotsubashi on Knowledge Management*. Singapore: John Wiley & Sons (Asia), 2004.

[4] J. Coplien, "There is no Kaizen without Hansei," J. Sutherland, Ed. Somerville, MA, 2011.
[5] D. K. Sobek and A. Smalley, *Understanding A3 Thinking: A Critical Component of Toyota's PDCA Management System*: CRC Press, 2008.
[6] D. Pink, *Drive: The Surprising Truth About What Motivates Us*: Riverhead Hardcover, 2009.