Ken Schwaber's Blog: Telling It Like It Is

Waterfall, Lean/Kanban, and Scrum

June 10, 2010 Categories Kanban, Lean **69** Comments

There are four identified problem spaces: Simple, Complicated, Complex, and Chaotic. The identification of the problem space arises from the characteristics of the problem domain. In software development, these are the technology, the requirements, and the people doing the work.

If these characteristics, or dimensions, of the problem space are simple and well-known, the problem is defined as Simple. If the characteristics are unstable, frequently changing, and not well-known, the problem is defined as Chaotic. If the characteristics are not simple, but are more known and well-defined than not, the problem is defined as Complicated. Lastly, if the characteristics are not simple and are more unknown and undefined than otherwise, the problem is known as Complex.

Throughout the 1980's, I worked to make projects that used the waterfall process successful. Despite creating and revising methodologies that delineated waterfall, and even automating the underlying delivery mechanisms, I was not pleased with the results. Then, in the early 1990's, Jeff Sutherland and I formulated Scrum, an approach to software product management that was radically different from waterfall. We experienced success after success when we used it.

I needed to know why Scrum was so much more successful than waterfall. I had friends working on advanced process control at the DuPont Advanced Technology Center in Delaware. DuPont had mastered the production of simple polymers, like rayon, to the point where they could be produced anywhere in the world. However, advanced polymers like GoreTex was so complicated that it's production couldn't be distributed. My friends were working this problem. The head was Babatunde Ogunnaike (Tunde), one of the authors of the bible of the process control industry, "Process Dynamics, Modeling and Control."

I showed Tunde several waterfall processes, methodologies, that I had brought with me. One was my own, another was the Coopers & Lybrand Summit D methodology. Upon inspection, Tunde observed that the processes were very prescriptive, where there was a complete plan of how to turn raw materials into finished product. He also observed that the elements of the plan weren't well defined. The input, output, processes, and resources consumed at each point in the plan (task)

weren't of adequate detail to ensure correct flow. Tunde then asked me how successful these types of methodologies were. I told him that our project success rate was below 50% despite heroic efforts to improve and enforce the processes.

Tunde told me that defined processes, such as waterfall processes, are the basis of mass production. Although not the most cost-effective, they are dominant in volume. Experts lay out all of the workflow to turn raw material into finished products. They then hire managers to bring in resources to staff the various points in the workflow (automation is preferred for reliability, otherwise use people). The job of the manager and their supervisors was to teach the resources how to do the defined job. If they did their work correctly, the flow would be smooth and the predicted product would result. Sometimes the plan was wrong, and would be corrected. Most errors came from inadequate execution by the resources.

Tunde told me that the waterfall process aped this model. He said that this model was preferred, and always used as long as it was successful. If it weren't succesful, regardless of attempts to tighten it up and retrain the resources, there were other, more expensive ways to create the products. He asked me what the success, or yield, was. I told him that it was less than 50%. He was aghast. He said that in modern industry a yield of less than 99.7 was considered inadequate. Fifty percent yield would be like GM throwing out every other car it built.

Tunde suggested we look at other models. First, he suggested Lean, the domain of problems where there is some complexity, but there is more that is known and stable than there is that is unknown and unstable. Tunde said that this approach was similar to the defined approach, but that it took into account complexities that might lower the yield rate to an unacceptable level. First, experts lay out the line. Managers and supervisors hire the resources and train them. Then, the line is surrounded by metrics so the experts can monitor if perturbances, or unexpected events, are occurring. Whenever they are detected, the line is stopped, the perturbance smoothed out, and production resumed. The resources were also trained to spot unexpected complexities and had the authority to stop production until the complexity was resolved.

Lean processes optimized quality and value. Sources of defects were rapidly detected and removed. Everything that wasn't geared toward optimizing value was also removed. Waste was continually detected and removed. Decisions were delayed until the last moment to remove waste. Lean processes are the most cost-effective.

Tunde said that this approach fell apart when unexpected events overwhelmed the productivity of the line. More time was spent detecting and fixing complexities than building product. More time was spent inventing new metrics to detect complexities than was spent designing new products. He also pointed out that the Lean realm was still mass product development. It was hard to justify building production lines with controlling metrics for a run of just one product, or one release of software.

Tunde then took me to the realm of empirical process control, a technique used for small batch production when there is more unknown than known, where the complexity is greater than the predictable. He felt that this was more relevant to the field of software development. The churn in our requirements, the complexity of the many technologies, and the use of creative thinking by people made this approach most appropriate. This approach was set up to expect the unexpected through frequent inspection of progress and subsequent adaptation of next steps to optimize output. The basis of it was the container.

A container is a closed space where things can get done, regardless of the overall complexity of the problem. In the case of Scrum, a container is a Sprint, an interation. We put people with all the skills needed to solve the problem in the container. We put the highest value problems to be solved into the container. Then we protect the container from any outside disturbances while the people attempt to bring the problem to a solution. We control the container by time-boxing the length of time that we allow the problem to be worked on. We let the people select problems of a size that can be brought to fruition during the time-box. At the end of the time-box, we open the container and inspect the results. We then reset the container (adaptation) for the next time-box. By frequently replanning and shifting our work, we are able to optimize value.

There is a radical shift from prescriptive approaches (waterfall, Lean, Kanban) to empirical processes (Scrum). An organizational culture changes if required.

Prescriptive approaches rely on the ultimate ability to effectively plan and control the plan to success. Experts set up the work, managers staff and manage resources to do the work, and workers (interchangeable) do the work. Command and control optimizes productivity in predictive processes. Empirical processes rely on containers to control risk and to continually aim the results toward the highest possible value. Managers work is to pose the highest value problems, and to do anything they can to assist the people who are solving the problems. Creativity and collaboration are the hallmarks of this process.

All organizations have issues adopting Scrum. They are shifting from a prescriptive, manufacturing based approach to an agile, empirical approach. Every discrepancy between their current organizational culture and Scrum's value driven approach is highlighted as an impediment to optimizing value. The organization is then asked to change its current culture to achieve this agility and value. When the change is too hard, they often change Scrum and keep the impediment. We call these "ScrumButs" because they are said as, "We use Scrum, BUT we found that it was very hard to have daily Scrums, so we only have them once a week."

I was told that Kanban is frequently used when an organization cannot readily adopt Scrum. Many of Scrum most difficult aspects are then sidestepped. Managers are still in charge of telling people what to do. People can be interrupted at any time. People are still work in functional silos, preserving the jobs of functional managers. People are not allowed to work in containers, sharing skills and knowledge to bring complexity into solutions – instead they are worked on a pull (more sophisticated than push) production line.

Lean is more productive than waterfall. Its transparencies and metrics allow frequent adjustments to optimize productivity. However, that optimized productivity is creativity when people are used to solve complex problems. People are worked like cogs in a machine, and their work is optimized at the bit level, rather than being aggregated into the value level.

God help us. People found ways to have slack in waterfall, to rest and be creative. With Lean and Kanban, those hiding places are removed. We now have a progressive death march without pause.

Scrum is intended to be a place where enthusiastic people can work closely with their customers to derive the most valuable, creative products possible. Getting to this point is a hard road for organizations transforming from a waterfall, command and control, prescriptive culture. However the rewards are great and the goal compelling. Organizations that make this transition will outcompete everyone else and be places where everyone wants to work.

Those who cannot Scrum and slide back into the slickly branded ScrumBan and its like will find themselves becoming coal mines, quickly generating commodity products from a work force that cannot wait to unionize itself so it has more rest breaks.

69 thoughts on "Waterfall, Lean/Kanban, and Scrum"

1. Marcelo Lopez *says:*
   June 11, 2010 at 12:34 am
   Reply
   Ken,

   With regards to the last two sentence in the first paragraph…

   "If the characteristics are not simple, but are more known and well-defined than not, the problem is defined as Complicated. Lastly, if the characteristics are not simple and are more unknown and undefined than otherwise, the problem is known as Complicated."

   Did you not mean to say "Complex" instead of "Complicated" in the second to last sentence?

   - kenschwaber *says:*
     June 11, 2010 at 9:30 am
     Reply
     Yes, I did. I will fix that. Thank you for pointing it out.

     - Jason Reid *says:*
       June 11, 2010 at 9:56 am
       Hi Ken,

       I think it would help if you linked to a reference for these problem spaces (the Cynefin framework, I assume?). In their terms, I think you still have Complex / Complicated backwards, but I'm not an expert!

     - H.-P. Korn *says:*
       February 16, 2013 at 10:34 am
       @ Jaso:
       I also assume that this is based on Dave Snowden's Cynefin – wich (important!!) is a "sensemaking" and not a categorizing framework. How to apply this "sensemaking" is described in the section "Contextualization" in Kurtz, C. & Snowden, D. 2003, The New Dynamics of Strategy: Sense-making in a Complex-Complicated World, IBM Systems Journal, vol. 42, no. 3, pp. 462-83.
       To see it as a sensemaking framework is very important – also to understand why we use "agile" principles:
       We work "agile" (in the meaning of "empirical process control" and "incrementally-adaptive") in "complex" situations. BUT: To see situations as "complex" or "complicated" or "simple" is our personal and social "construction". Not the situation or a system is "complex" … we label them as "complex" … and we do it based on the

conversation in that social system we are part of. Supporting such a conversation is Snowden's "Contextualization".
That means:
The outcome of such a conversation my be, that in a specific situation "the relationship between cause and effect requires analysis or some other form of investigation and/or the application of expert knowledge, the approach is to Sense – Analyze – Respond and we can apply good practice" (referencing Cynefin). Then something like a "plan driven approach" will be appropriate.
If the shared understanding created in such a conversation is, that in a specific situation "the relationship between cause and effect can only be perceived in retrospect, but not in advance, the approach is to Probe – Sense – Respond and we can sense emergent practice." then an approach based on "empirical process control" and "incrementally-adaptive" will be appropriate.

So:
Without having such a conversation for contextualization first a discussion if a plan driven approach or an "agile" approach is "better" will end in something like a "battle of religions"….
.

2. Pawel Brodzinski *says:*
June 11, 2010 at 2:42 am
Reply
You look at different methods from very high level, basing on what they were intended to be. At the same time I don't agree with your view of lean but that's a different story – I'm not going to try to convince you.

The problem I have with your approach is that it is pretty academical. You try to look at the big picture only, while most agile teams are small, fewer than 10 people. It doesn't really matter if they work with Scrum, Kanban or waterfall – you shouldn't look at this bunch of people as you look at factory.

Believe it or not, but teams, Scrum teams included, usually struggle with surrounding reality, they don't consider themselves as creative black boxes isolated from their managers. What more Scrum, more often than not, is implemented using bottom-up, not top-down approach. This means proper support from senior management is often a problem. This also means that people don't look at Scrum teams the way you do, trying to let the work for as long as iteration lasts.

Besides, business often enforce priority changes and teams just can't work in complete isolation.

And there is one thing more – I don't believe there is a silver bullet and Scrum isn't one. I believe, depending on an environment you work in, depending on people you work with, different flavors of different methods would work for you.

With all that, I find it impossible to agree that there's one way we should align to. And all ScrumButs, ScrumBans etc are just different flavors of similar ideas. What more, I find Scrum and Kanban basing on the same principles and so do many people I work with and I talk with. After all Scrum, Kanban or waterfall is just a tool. The real value, creativity and productivity

come from people, not from method. Get a small integrated team of great engineers and they would deliver no matter whether they follow Scrum, Kanban or waterfall. Get a bunch of losers and Scrum won't help them.

- Ben Hughes *says:*
  June 14, 2010 at 2:37 am
  Reply
  Pawel,

  The reality is now changing (at least here in the UK) regarding bottom up adoption. Increasingly now, we are seeing Scrum being adopted by organisations from CIO/CTO down, but usually having diminished expectations of just how much change is required. Usually, often quickly, politics overrides the value proposition, in which case it is completely valid to draw the manufacturing comparison – realistically at that level software teams are seen as resources rather than people.

  It is exactly the point of side stepping the real issues (taking politics over value), that keeps the organisation in transitional mediocrity, rather than gaining competitive advantage.

- tobiasmayer *says:*
  June 17, 2010 at 1:39 am
  Reply
  @Pawel

  > After all Scrum, Kanban or waterfall is just a tool.

  Scrum is not a tool. It is an enormous mistake to think of it as such. Scrum is a framework for change, it is a way of being, a philosophy, a mindshift. To miss that is to completely miss the point of Scrum. It is no wonder Scrum implementations fail so often if this is the common mindset. Scrum requires you to question /everything/. All old assumptions like "we need tools" need to be challenged. We don't need tools, we need hearts and spirits, we need passion. Scrum offers a new pathway. Few dare walk it.

  - Matthias Bohlen *says:*
    June 17, 2010 at 11:02 am
    >we need hearts and spirits, we need passion

    Exactly! That's how I always try to work, in every environment. This is independent of the method I use. I enjoy the kick when the team becomes creative and hyper-productive.

    What's so difficult and (at first sight) confusing about Kanban is that it is not a method. It has no roles, no activities, no result types, no nothing. Just a WIP-limited pull system. Kanban is always based on an underlying way of working: If the underlying is Scrum, then you get Scrumban. If the underlying is waterfall, then you get Waterfallban, etc.

    Kanban is like a spice, not like a meal. It's like salt or chili pepper. You can't compare it to potatoes or meat. That makes it difficult to understand.

    Example: Tobias, you ask me: "@Matthias what is a "piece of work?" If a team of 7 is working on 9 things that surely implies it is at the task ("How") level. But tasks have no business value."

This is a question that applies to the underlying, not to Kanban. Kanban simply manages work items and does not prescribe what a work item is. The team has to use the underlying to find a definition of what a work item is or what "done" means.

Hope, this does not add to the confusion.

My 2 cents,
Matthias

- Eric Willeke *says:*
  June 17, 2010 at 12:23 pm
  s/Scrum/Lean
  Agreed! Tobias, when you talk about Scrum, it often sounds very little different than how many (most?) within the lean/kanban community think about lean.

  When people ask for tools, they often seem to need new perspectives. In many cases, teams will evolve effectively towards a lean/kanban system as an emergent behavior with just a few of the key agile values inspired. Things we've been doing for years like "focus", "respect", "trust". Simple statements like "Stop starting and start finishing" or "Don't start what you can't finish"

  I tend to belive what puts many people off on lean/kanban as a "rigouous methodology" or a "prescriptive technique" or "just a tool" is that we excitedly and passionately discuss the journies we're in the midst of as if they are a destination. We gather, write, and share about "where we are now" and revel in the fact that so many of our paths seem to have passed the same place, which leads us to talk about the tools we used to get there, the things we saw along the way, and unfortunately, even brag about how much better we covered the path than somebody else. At times I feel the same way about Scrum, yet I then get inspired when I hear people speak as you tend to.

- Alan Shalloway *says:*
  June 27, 2010 at 10:26 pm
  Well, we agree on this. I've never thought of Scrum as a tool either. Unfortunately, many dare walk it, and find it insufficient. To say when it isn't it's the fault of the walker both disrespects the walker and hides any holes in the framework.

3. Pingback: *funkybrain » Ken on Waterfall, Lean/Kanban, and Scrum*

4. Alan Shalloway *says:*
   June 11, 2010 at 8:51 am
   Reply
   I would highly suggest reading this:
   Types of Processes by Don Reinertsen.
   http://www.netobjectives.com/blogs/Types-of-Processes

   - Paul Beckford *says:*
     June 16, 2010 at 12:56 pm
     Reply
     Hi Alan,

I think Don and yourself have missed the point. In Cynefin terms the distinction of interest is ordered versus unordered systems. I can use feedback and the system still be ordered.

Unordered means that the outcome is no longer predictable in advance. There is no formula that can be used to predict outcomes. No nice neat equations (Second order differential equations ring a bell from my dim and distant electronics past)

So what happens in the unordered complex space is that intelligent agents within the system – namely people, perform ad-hoc adjustments, using a probe-sense-respond approach to keep the desired outcome on track. No one can predict the likely outcome of these adjustments. You can only make sense of them in retrospect.

This is very different from feedback in say electronic systems.

Regards,

Paul.

5. Ernie *says:*
June 11, 2010 at 12:18 pm
Reply
Is it possible that you are mischaracterizing the outputs of a software team? You state:

"It was hard to justify building production lines with controlling metrics for a run of just one product, or one release of software."

But the output of the 'production team' isn't Product A or Product B. It is more abstract. E.g., A software fix, or an implemented user story/requirement, etc. It shouldn't matter what the domain is, so long as the tasks are well understood.

And if the tasks are not well understood? Well, I don't see how a particular methodology is going to help with that. There is a difference between the complexity in the problem domain, e.g., "what are the legal regulations for this use case" and complexity in the process model.

Finally, if you have a low maturity organization, or a poor manager, or no corporate support, it doesn't really matter what process you adopt. I can't see anything working in that environment, although Scrum as disruptive innovation might be the best way to start.

6. Pingback: *Kanban is a Gateway Drug*

7. Mark Kennaley *says:*
June 11, 2010 at 2:23 pm
Reply
Two things:

1) You seem to not understand how to apply feature crews as self-organizating, cross-functional teams in an Acceptance-Test Driven Development A-Plant configuration, pulling from the same fine-grained demand management queue as in Scrum with utilization below critical WIP;

2) You seem to reference Classical Control Theory and/or Modern Control theory, and yet preclude the identification of Adaptive Control Theory and gain scheduling / CAS plant reconfiguration due to changing dynamics within project flight – an outright acceptance of sub-optimal delivery dynamics and Integral of Absolute Value of Error – are you a control engineer?

If you read my recent works, you will see a concrete basis for why you need both Scrum during the transient period of time of dynamic system response, and JIT leveraging Kanban at a fine-grained demand level (which has nothing to do with some sort of implication of prescriptive, serial manufacturing process) for scale past the exercising of real-options on high-stakes design decisions. You can also see what I am talking about in position paper at http://www.semat.org/pub/Main/ZurichWorkshopProceedings/04_Kennaley_position.pdf

Cheers.

- Matthias Bohlen *says:*
  June 13, 2010 at 11:16 am
  Reply
  Mark, to whom were you responding? Which one of the persons do you mean by the word "you"?

  - Mark Kennaley *says:*
    June 13, 2010 at 6:01 pm
    Matthias – the comment was meant for Ken – it was his posting. I am curious with all the talk of Control Theory & other sub-branches of General Systems Theory whether he has the depth of experience to make such representations. I ask because I am an EE, and have recently applied this body of knowledge to explain why Agile works, beyond a tacit or superstitious basis – or a superficial one. My understanding is that folks have historically only thrown these terms about, but haven't put the time in and all the math. I aim to change that. And specifically, it is pretty obvious how Scrum and Kanban relate with respect to Adaptive Control theory – I wonder why all the rhetoric?

8. Matthias Bohlen *says:*
   June 11, 2010 at 6:38 pm
   Reply
   Nice article, Ken! Especially the part that explains what a container is and how it is supposed to work! I have realized today that there is more to an iteration than just a timebox.

   However, as a Kanban practitioner in software development, I'd like to point out one thing: Kanban is not a "prescriptive approach" as you call it. It is neither prescriptive nor non-prescriptive, it simply displays a number of things:

   * what kind of work people do (a workflow)
   * how much they do at the same time (WIP = work in progress) and
   * in which state of the workflow each piece of work currently is.

   After displaying this, Kanban now simply asks the team members to respect a so-called "WIP limit" that allows them to focus on their work without being distracted by ever-changing priorities. For example, a 7 person team might agree to adopt a WIP limit of 9 pieces of work, that means, they will never work on more than 9 pieces at a time (7 in progress and maybe up to 2 blocked pieces because questions have to be clarified).

When a team member is done with a piece of work, he/she pulls another piece onto his/her desk.

And that's it! There is no such thing as a "progressive death march without pause". People can go for a pause and have coffee whenever they feel the need. They rest whenever they decide to do it. Kanban does not exploit people more than any other method because Kanban is no method – the work is done using the existing method that the team has in place. Kanban is something that makes the work more transparent but it is not a method in itself. It is a non-method, if you want (at least, that's the way I see it).

Kanban is the opposite of a "progressive death march without pause". Imagine a team of acceptance testers – its duty is to help the product owner find out whether the backlog items have been implemented correctly. What happens if the entire test team gets the flu? The developer teams have a WIP limit that is now reached because the testers are sick and do not pull readily developed stuff away from development into acceptance test. So, stuff remains "in development".

Now what next? The developers respect their own WIP limit and stop developing – they take a rest! They read books, go for a training, refactor their software, learn something new that they always wanted to learn, and so on… When the testers recover from the flu and get back to work, they will find that there is no additional pile of work created by the developers in the meantime! No "death march" at all – they simply resume their work and pull items away from the developers. And, the developers resume their work, the line starts to flow again.

Kanban in software development is very much different from mass production. I invite you to join the Yahoo group called "kanbandev" where the Kanban community actively discusses these things. I'd really like to see your comments on their messages in the Yahoo group, and I will be excited to read what you have to say.

Cheers
Matthias Bohlen
CSM and Kanban practitioner at the same time

- Wolfgang Frank *says:*
  June 12, 2010 at 6:49 pm
  Reply
  Matthias, I totally agree with you! I have done several Scrum projects without the ScrumButs and I've seen organizations that were not able to adapt to Scrum due to several reasons … Anyway they have to do their job and are nevertheless open and motivated … I think this has to be respected and a suitable way to provide the best possible results has to be found. For me applying ideas and tools like Kanban (in a way Matthias describes it) worked best for me and created results that were not better or worse than by applying Scrum. I like to use Scrum, but it is not the silver bullet Ken thinks it is! When I need to drive a screw into a wall, maybe a Kanban screwdriver is better suited than using a Scrum hammer – that might

- tobiasmayer *says:*
  June 16, 2010 at 4:14 pm
  Reply
  > Now what next? The developers respect their own WIP limit and stop developing – they take a rest!

Resting is good, of course. But sometimes there are things that need to be done, maybe even commitments to keep . If the team worked cross-functionally the developers could (crazy idea) do some testing.

One of the problems I have always had with the Kanban-type approach is it seems to not only maintain, but promote silos. The Scrum-type approach promotes teams, shared knowledge and mutual support. Development does not need to stop when one or more team members get sick.

Imagine applying that approach at home. Only one person washes up, another takes out the garbage, etc. Washup person takes a week vacation. No one "upstream" is allowed to eat for a week. Garbage boy gets sick for a few days. Smelly house. Doesn't such an approach seem insane? Why use it at work?

- Eric Willeke *says:*
  June 17, 2010 at 12:42 pm
  "One of the problems I have always had with the Kanban-type approach is it seems to not only maintain, but promote silos."

  I have observed a natural tendency for teams to evolve towards cross-functional ("Generalizing specialists") in kanban teams that are allowed to evolve. In the language we tend to use, the emergent behaviors of the system is to self-optimize by reducing the length of the value stream by minimizing handoffs. In my preferred langauge, it means that people are leaning to speak each others' languges and becoming more able to perform cross-functionally, allowing the system to change in reaction to the increase capabilities.

  Unfortunately, there are often external impediments to break down in this process. Things like the incentive model used by an off-shore group. Things like strongly siloed matrix management. These issues may carry more inertia, but they can still be redirected if enough people work together to do so.

  Happily, this breakdown of silos isn't constrained within the teams. I've seen team members pull in (and go out to) work more effectively with deployment and operations. People start sitting with customers and working directly with them. People become more accurately aware of the core business metrics and align their behaviors accordingly. Both Scrum and kanban allows this behavior to emerge, but how effectively do the different mindsets actively encourage it?

- Lonny Eachus *says:*
  August 17, 2010 at 4:22 pm
  Reply
  The problem here is that Ken was referring to Lean/Kanban in a role similar to the way they are used in manufacturing, which is definitely a prescriptive approach. But the way Lean and Kanban are used for DEVELOPMENT is far different. So different, in fact, that it is rather difficult to compare them in these disparate roles, and Ken is 100% wrong to apply them as though they worked the same in both situations. The very idea is ridiculous.

  - Ken Schwaber *says:*
    August 19, 2010 at 2:15 pm

Kanban is fine if the Team chooses to use it as its own technique for managing itself. It is wrong if it is imposed by anyone else as the technique to use, regardless.
Wow, I've been 110% wrong before, but never 100%.
Ken

- Ken Schwaber *says:*
  August 18, 2010 at 3:35 pm
  Reply
  If Kanban were a technique chosen and used by the Team (of developers) on a Scrum Team to manage themselves, I'm all for it. However, it is almost always used and discussed in conferences as a technique for management to manage the self-organizing team to get it to be better managed and provide more metrics on how to improve it.

  - Eric Willeke *says:*
    August 18, 2010 at 3:42 pm
    In my experiences, Kanban adoption is often intertwined with the collective realization that there is no line between management and "the team" in a given organization. The total transparency and shared control that arises quickly makes the source of impetus for the change irrelevent as the team-driven improvments accelerate. As an aside, are the conferences you mention fairly recent? (Last year?) I believe the message and perspective of the community has changed drastically starting about the time of the Lean and Kanban conference in Miami about 18 months ago. Thank you!

9. Vic Williams *says:*
   June 12, 2010 at 11:15 am
   Reply
   Excellent, as expected. The container explanation is great. Other comments here hint to me that belief systems are threatened by scrum and your writing. It would be interesting to test critics somehow to see if "self-empowered teams" are killed off with some such ways of thinking. Thus showing the need for skunkworks and explaining higher than needed scrum failure rates.

   I thinking scrum/agile has aspects pointing to better education for a post-factory society, and factory-thinkers are retarding the necessary changes.

   I suggest that a skunkworks is a design-thinking-doing model and scrum almost accidentally falls into design-thinking-doing as the team develops.

10. tobiasmayer *says:*
    June 16, 2010 at 4:26 pm
    Reply
    Nice post Ken. You express some of my own concerns with the lean/kanban/hybrid approaches many are turning to. I also think it is often a cop-out, a way to maintain the status quo while showing that Agile "works", i.e. speeds things up and removes waste.

    But there is so much more to solving complex problems and creating joyous work spaces that these two things. The team-interaction element, the collaboration and self-organizing that Scrum promotes seems to be missing in most discussion on Kanban, as does any talk of human values. And this:

"Tunde said that this approach fell apart when unexpected events overwhelmed the productivity of the line. More time was spent detecting and fixing complexities than building product. More time was spent inventing new metrics to detect complexities than was spent designing new products."

Exactly. An over-focus on process is only ever a short-term solution. Over time it is more likely to stifle creativity than foster it.

1. Pingback: *Links from Twitter - Vagif Abilov's blog on .NET*

2. Andy Roberts *says:*
June 17, 2010 at 3:55 am
Reply
@tobiasmayer . There's a nice cartoon illustrating what happens when the garbage boy gets sick for a few days.

http://blog.crisp.se/henrikkniberg/2009/06/26/1246053060000.html

Also it's worth checking out comments from Matthias above. While I learnt a lot from Ken's blog post, unfortunately some of his reasoning about Kanban appears to be based on misinformation about Kanban, which Matthias explains.

- tobiasmayer *says:*
June 17, 2010 at 9:12 am
Reply
Henrik's cartoon description is excellent. I'd add though that this is how good Scrum is practiced: a focus on one story at a time, with the whole team swarming on the tasks for that story, and supporting one another until the story is complete. WIP limits are not imposed, but agreed by the team — sometimes it makes sense to work on 2 or 3 stories at one time.

  Kanban has always appeared as a more silo'd approach (and Henrik's cartoon expresses that), not encouraging whole-team commitment from the start, but only in times of crisis. Many Kanban boards call out specific roles (e.g. design, dev, test).

  The other aspect of Scrum that I feel is particularly powerful, and lacking in any Kanban description I have read is the important separation between What and How. E.g. @Matthias what is a "piece of work?" If a team of 7 is working on 9 things that surely implies it is at the task ("How") level. But tasks have no business value. Where is the WIP on the "What"? Without that distinction we lose sight of the bigger picture. Limiting tasks doesn't really mean very much.

  > When a team member is done with a piece of work, he/she pulls another piece onto his/her desk.

  What does "done" mean in this context? Again we have a sense of a silo'd operation, and old-thinking, as in "I've done my part".

  - Andy Roberts *says:*
  June 17, 2010 at 3:19 pm
  The columns on the Kanban board represent activities. Kanban is agnostic as to whether those activities are performed by specialised teams or a single cross-functional team.

In kanban (at least as practised here) WIP limits are determined empirically. We have arrived at point where we know a certain number on each column results in an even flow. So WIP limits are not really imposed (at least, not here), they are worked out, with the team, over time, with the aim of balancing the flow.

You'll get the WIP on 'what' when more of the whole value stream (from concept to revenue generation versus the actual production of code) is mapped on the Kanban board. Again, Kanban mandates nothing, but it's not uncommon to find kanban implementations that in the early part of the value stream, deal in large units of work (a 'story' or 'minimum marketable feature set') which will have business value, but at the point in the value stream where developers begin to work, there's a smaller unit of work.

Not 100% with you on the "I've done my part" thing. Are we talking about a dysfunctional individual, no team spirit, doing things by the book ? They'll cause problems anywhere, won't they, in Kanban, Scrum, or Waterfall ? The method (either Kanban or Scrum) will most likely surface that dysfunction to be dealt with.

13. Juan Bernabó *says:*
June 17, 2010 at 8:16 am
Reply
Hi,

Great article.

One concern that I have is that the fragmentation, and little differentiations that are occurring today in this field, to be honest, in my humble opinion, have little to do with people looking for solutions to problems they have, but for consultants trying to establish a differentiator, which is a problem but for the methodologists/consultants and not for the users. This approach buy us more confusion in the field.

We probably have to reflexion on why, and what for, we need this little differentiators, if we are really solving an user problem or just creating more confusion make it more difficult for the people that needs those solutions to choose from.

I use a lot of lean thinking (not lean practices), toc, empiric process control concepts to help people understand at high level why Scrum works so well.

Having people experimenting different things is great, but what changes everything is what is the motivator, why they are doing that, what is the problem they are trying to solve, and sometimes they are simply doing those things to foster a marketing pitch or create a new differentiator for their offer, which may not really add value to the real customers.

Juan.

14. Pingback: *Fridays Digest #18 Scrum vs. Kanban | Edge of Chaos | Agile Development Blog*

    - *kenschwaber says:*
    June 17, 2010 at 2:14 pm
    Reply
    Rejecting this trackback per comments policy

15. Peter Green *says:*
June 17, 2010 at 1:20 pm
Reply
Ken, thanks for posting all of the details about the DuPont story, I had always wondered about the specifics of that experience.

I think that Lean might start to provide appropriate solutions outside of the scrum team at the business operations level, but not necessarily at the development team level. I know that one of the challenges to scrum from the Lean/Kanban folks is that a scrum team can become a sub-optimization within the larger business, and I have seen this in my org where the scrum team by increasing their velocity has now outpaced the ability of the business to take advantage of their productivity. This requires changes outside of the scrum team, and lean can help to identify where the business bottlenecks are.

At the team level, Lean or TOC are just another set of ideas that a self managing team may draw from in the process of inspecting and adapting. By asking teams to figure out how to get something all the way done ("through the line") in a sprint, and then inspecting the results of that work at the end, they may realize that, for example, they are consistently running out of time to test (QA is a bottleneck station), and so the dev specialists within the scrum team might need to help out with testing towards the end of a sprint, or they might need to use better early defect removal techniques, or they might use some of their excess hours to build a better automation system. They don't just stop working, they swarm around the problem and solve it.

16. Pingback: *Slack time at Mark Needham*

17. Pingback: *Slack time | ASP.NETer*

18. Rodrigo Dellacqua *says:*
June 19, 2010 at 3:10 pm
Reply
Ken,

As a Project Manager at IBM, where they use their own definition of Waterfall, the main problem I see when managers talk about implementing agile and how hard it is, is due to the fact of the complexity scale you mentioned.

Scrum isn't for everything, that what most people don't get. You don't need Scrum to install X number of servers and Deploy X number of *nix OSes, that's a very known and simple problem.

What I see the most is managers that come from managing people that work with Simple problems (Infrastructure, Telecom, Production) , to come manage a project with complex problems, you also need to manage people but that's secondary.

They so blindly thinks that what they used their whole life can be applied in every scenario.

Very thoughtful post.

- Charles Bradley *says:*
June 22, 2010 at 3:08 pm
Reply
Rodrigo,

I see the same thing all the time. My favorite version of this is when companies have highly matrixed teams and try to use Scrum. I tell them flat out…it won't work, and you should find a different process or process framework.

Charles

19. Pingback: *Oneda | Aconteceu no Twitter 21 - 13/06/10 a 19/06/10*

20. Charles Bradley *says:*
June 22, 2010 at 3:05 pm
Reply
Ken/All,

As a Scrum Coach and ScrumMaster, I've seen the exact same thing and it has troubled me for some time now. At best, it is poorly implemented Agile. At worst, it's downright fraudulent.

To me, the Agile manifesto, Kanban, Scrum-bahn, and Lean, due to their very conceptual view of software development, are all open to interpretation, mis-interpretation, dis-information, abuse, and fraud. Further, structural components within some of these frameworks lend themselves to abuse and fraud. My experiences tell me that the large majority of people who use these frameworks, without proper education(self or otherwise), are more likely to hurt themselves and their organization than help them. Sad but true, at least IMO.

Scrum does a much better job at preventing this kind of abuse, as does XP. They do so with much more practical/tactical level rules and practices. Don't get me wrong, Scrum and XP are also open to the same kinds of abuses, just much less so. I have my own fears about a small number of concepts even as written in the Scrum Guide.

In the hands of process experts, probably all of these frameworks have a chance to succeed. However, the number of process experts in the industry right now does not allow for continued and sustained success.

Anyone can talk "Agile," but very few can BE Agile, and even fewer can shepherd Agile in any sustainable and successful way. The more practical the framework, the more sustainable it will be.

As such, the ones mentioned above and in Ken's blog post are not at this time sustainable IMO.

- Zach B *says:*
March 30, 2011 at 6:31 pm
Reply
"…more likely to hurt themselves and their organization than help them. Sad but true, at least"

Agreed.. Education and awareness of the potential problems of not implementing the scrum methodologies correctly should be heightened.

21. Peter Merrick *says:*
June 24, 2010 at 6:21 am
Reply

What I like about Scrum is, in the right circumstances, it works and software gets delivered. That's a big tick in the box. If I know I can deliver software, I can focus on my requirements. If I can make the requirements of high quality then I know the team can deliver against them. For me the hardest part of the delivery process has always been getting the requirements right (user stories if you will). Everybody has a story to tell, and that's a good thing, but some stories are better than others. If the backlog is 'ungroomed' then 'pulling' them into a planning meeting is going to be inefficient, and inefficiency is waste and waste is to be eliminated. So thanks to Scrum, we can focus on requirements where before we couldn't focus on anything because it was so unclear as to where the failure really was. I'm an analyst, and I'd like to see a new role in Scrum called master story teller. After all requirements (features – whatever; language is slippery) are really just a reflection of the clarity of the thinking of the business. And the clarity of the thinking of the business is a reflection of the clarity of the organisation's enterprise goals. I'm pleased Scrum has such a strong pedigree in system's thinking, but everything must evolve to fit the need to which it is put. In all things, evolution is preferable to revolution. Organisations are risk adverse so there is bound to be a roadmap to Scrum adoption. Maybe that includes Kanban; why not? If somebody needs it, gets something out of it then put it in. Which leads me to the point of 'pulling' stories/tasks into the Sprint. It's only going to work with great stories that are comparable as apples with apples not apples with orchards. Otherwise the Kanban pulls in rotten fruit.

22. Damon Poole *says:*
June 24, 2010 at 3:30 pm
Reply
Hi,

This discussion seems to crop up on a regular basis on various blogs in various forms.

Here are some observations:

1) Having the Agile leadership squabble over "what is Agile" seems to me to be more harmful to those that are new to Agile than it is useful.

2) In reading the Agile manifesto, I see no mention of Scrum, Kanban, XP, DSDM, FDD. Agile is more than any of these, isn't it?

3) If Agile is about uncovering better ways of developing software, aren't assertions along the lines of "all you need is Scrum" a contradiction?

Please note, I love Scrum and I have a great deal of respect for the folks that wrote the Agile Manifesto, but I strongly believe that Agile is bigger and more important than any of us or any defined thing. I say thing because I'm tired of the endless this is not a methodology and that is not a process.

Cheers,

Damon

23. Steve Zhang *says:*
June 25, 2010 at 11:22 pm
Reply

Hi, I am new for Scrum. I am a developer, it seems for me Scrum is a tool for managers instead of developer side, because it does not mention to guide how developer should work. there is no TDD, no refactoring, no pair programming, and managers does not care how developer finish the job, it seems Scrum manager only care about the result. Does it mean implementing the Scrum has a much high requirement for the developer so manager trust their skills and just put them into a black box then wait for the output? But in reality what if a team the developers don't follow TDD, and refactoring , how to guarantee the quality using Scrum?

- Ken Schwaber *says:*
  July 7, 2010 at 2:53 pm
  Reply
  We are trying. Look at the Scrum Developer program on Scrum.org, which intends to improve this hole that XP has left.

  - Matt Baker *says:*
    July 9, 2010 at 3:58 pm
    Do you mean "intend to improve this hole with XP"? It has been my experience that it is hard to be successful with Scrum unless the product team also adopts XP.

24. Edwin Dando *says:*
    July 14, 2010 at 9:09 pm
    Reply
    Thank you for an extremely interesting and thought provoking article. As a Scrum Coach I do see a place for Kanban but think the use of it needs to be carefully considered.

    Scrum is hard – extremely hard. I discussed this with Jeff Sutherland earlier this year and he re-iterated the "no silver bullet" concept. He said to simply consider a scale of low productivity versus hyper-productivity (i.e. around 500%). If you want to move up the scale to the hyper-productivity end then you need to implement Scrum properly and completely along with continuous integration, TDD, BDD (which aren't part of Scrum)etc. Every time you do a "Scrumbut" (i.e. we don't use retrospectives, we can't get a decent product owner, we don't have a test environment) you simply restrict your ability to move up that scale. It is as black and white as that.

    From implementing Scrum at many clients here in New Zealand I see some (most to be frank) really struggle with the shear level of change. The difficulty tends to come from functional managers and project managers who are indoctrinated in the concept of specifying everything up front, delegating work out to subordinates and then managing via central, hierarchal command and control. The concept of emergent requirements & design and self-organising teams tends to challenge the fundamental need for their position/career and in my experience they struggle to adapt to this new paradigm.

    They then embrace Kanban as they are seen as embracing the new world, it is better than what they do now and it is a lot easier to implement (and less threatening) than Scum. It is seen as an evolution not a revolution and can simply be dropped on top of an existing methodology. Managers then get more visibility, however this is often then used for further centralised decision making and control. They fail to grasp the core philosophy (and benefits) of Scrum and only manage a slight shift up up the hyper-productivity scale.

I can fully understand the desire to split up the "In Progress" section on a Scrum board into sub-sections (for example analysis, design ,development, test etc) with WIP limits. It provides finer grained visibility as to where exactly a task is and what is holding it up. However complex problems require a mixture of creative and scientific thought. Restricting team members to just a specific role ala the production line compromises this.

I prefer to have team members consider multiple perspectives on a problem and work creatively across different roles. For example, we teach developers how to improve code quality by writing their software to be testable. To do this we ensure they spend time working as a tester in a team, thinking about overall product quality in design, architecture and implementation. They have to consider how to write code to make testers life easier. We teach testers testing by getting them to spend time writing requirements (user stories) and modelling business process so they have a sufficient base line and domain knowledge to guide the writing of good tests (usually unit tests). I appreciate that in Kanban tea members can be redeployed to other roles (for example moving an analyst over to do some development) but the core concept of problem solving via multiple perspectives still does not seem "native" to Kanban.

Implemented well I think Kanban can be extremely powerful, however why it is being adopted needs to be well considered. Is Kanban being adopted because it is the best approach to the target business environment at that given point in time or is it being adopted because the change associated to implementing Scrum requires too much work and is going to upset established paradigms too much?

I think Kanban can be a useful stepping stone towards full Scrum adoption if the business environment is too challenging to implement Scrum first off, or in an IT support type environment where issues need to flow through a system and be released immediately without waiting for an iteration to end.

For now I am going to hold fast to the philosophical core of Scrum but am watching the Kanban space with interest.

25. Pingback: *PMI San Diego Chapter | Home*

26. Victor Szalvay *says:*
August 10, 2010 at 1:42 pm
Reply
I applaud Ken's concise argument delineating Scrum from Lean/Kanban. In the software industry, I'm seeing a movement toward Lean/Kanban as the "next big thing" after Scrum. What people don't realize is that they are actually taking a step backwards in their process evolution (or "sliding" as Ken says).

I hope people read posts like these before jumping on bandwagons. Lean and Kanban seems to be driven in the market today by a class of organizations that were left out of the Scrum "trend". Unfortunately, new doesn't always mean better, as Ken points out above. Scrum is still the best way to approach complex software product development regardless of the latest marketing.

– Victor Szalvay

- tobiasmayer *says:*

August 10, 2010 at 1:59 pm
Reply
+1 to Victor.

> Scrum is still the best way to approach complex software product development regardless of the latest marketing.

I agree. But implementing Scrum takes a great deal of courage. The sad truth is that many are not up to the challenge, so are looking for short cuts. This is true as much of the coaches and trainers as of the organizations themselves.

27. Pingback: *links for 2010-08-12 « Dan Creswell's Linkblog*

28. P Strong *says:*
September 2, 2010 at 2:24 pm
Reply
Submitting a question from a team at work that is doing really well at Scrum transformation but needs an update from Ken's 2001/2002 book. Would like a response from Ken if possible as team would like to hear it from him:
"In your Agile Software Development with Scrum book from 2002, you say that sprints should be 30 calendar days. Is that still your current recommendation for the length of a sprint? We've also heard 2-4 week sprints, but wanted to get it directly from you, if possible. Any other details, such as when you recommend Sprint Planning Meetings, Review Meetings, etc. would be appreciated."

- Ken Schwaber *says:*
September 4, 2010 at 5:48 pm
Reply
Thirty days (one month) is the maximum length given the complexity of software development. Many organizations use two week Sprints, others one week sprints with success.
When devising your Sprint length there are several guidelines.
1. Try to keep it the same. If a football game is always the same length, the team learns how to do what within that duration. Also, the fans learns when and how to interact.
2. If there are more than one team working on the same product, they should all have a fixed point (monthly?) when their work integrates and is integration tested as a complete, "done" increment that the customer could potentially use.

  The four variables that I consider as relevant to the complexity of software development are the people engaged, the technology employed, the clarity and stability of the requirements, and the length of the iteration (time between empirical inspect and adapt points). If you rate people, technology and requirements from 1-9, 1 being simple, 9 being the most complex, multiple them together. Minimum 1, maximum 729. Now multiply this figure by the length of the Sprint, in days.

  If you are finding that the work is too complex, you have the option of decreasing any of the four variables. Find more experienced people. Use simpler technology. Reduce the length of the Sprint.

Best,
Ken

- Larry White *says:*
  May 27, 2011 at 8:02 pm
  Hi Ken, Fantastic article, but I'm less sure I completely buy into the four variables you mention: Technology is a variable that doesn't vary much – most of your readers (I'm guessing) are doing Mobile or Web. Experience, I was surprised to see in my own research, didn't have much impact. I think (totally unsupported hypothesis) that using fewer people would often be more effective than using more experienced people. Requirement quality and stability – sure, but it's another variable that is hard to vary. Sprint length I buy completely. I think, too, that small improvements in code quality – by whatever means – would pay big dividends over the life of a project. Cheers and keep up the great writing.

29. Pingback: *Sprints and Compelling Goals at Urban Turtle's blog*

30. Pingback: *Resumo rápido do evento pensando em Lean | Blog Lambda3*

31. Pingback: *Kanban in 24 Hours « Managing Software Development*

32. Pingback: *Hat Scrum etwas mit Kanban zu tun? | Komplexitätsmanagement*

33. Pingback: *Software Development Processes - Welcome - ThoughtWire*

34. Pingback: *Punctuated Equilibriums, Containers, all things Complexity and how Kanban fits in | Yeret on Agile/Kanban*

35. Pingback: *Bericht vom 11.01.2012 | Limited WIP Society Cologne*

36. Pingback: *Scrumin etuja Kanbaniin ja Scrum-baniin verrattuna « larelekman.com*

37. Lean Muscle Formula *says:*
    July 20, 2012 at 4:29 am
    Reply
    This piece of writing is actually a nice one it helps new net viewers, who are wishing for blogging.

38. Mike C *says:*
    November 29, 2012 at 2:42 pm
    Reply
    Although there are some good points here I fear that this whole "Scrum vs. Kanban" thing has degenerated into some kind of religious war. In the organizations I have been involved with, I have found that sometimes Scrum is appropriate, sometimes Kanban is more effective, and sometimes a hybrid approach works better, It all depends on the situation. For example, we sometimes switch from Scrum to Kanban during stabilization periods in lieu of having "stabilization sprints" of a fixed size. I have encountered other situations where judiciously applying WIP limits inside of the Scrum framework turned out to be very useful in correcting certain kinds of issues. In these particular cases, simply shortening the sprint would have been less useful.

Scrum can be a tremendous improvement for organizations using waterfall but it is far from a total solution and like all methodologies it has its own set of problems. Some of the most game changing improvements (continuous integration, automated testing, etc. have nothing to do with Scrum (these two come from XP) but I don't know of any organization that calls itself Agile that doesn't do these things. The best improvements come when an organization is able to look at the whole landscape of tools, methodologies, and practices and determine what's best in their situation.

I tend to disagree that Ken's characterization of Kanban as ".. a progressive death march without pause". Most organizations in the local community here to not run their projects this way and I would like to mention that ANY methodology, Scrum included, can be abused.

The local Agile community here in Austin have very strong Lean/Kanban proponents who now argue against Scrum with the same level of disrespect for alternative views that I see on this post. Zealots on both sides of the discussion need to realize that this kind of rhetoric is counterproductive and that we need to be talking about tools and techniques to help us all improve instead of dividing into camps based on ideological zeal and/or allegiance to a particular brand. It has gotten so bad here that we now have separate "Agile" and "Lean" organizations that generally don't talk to each other.

I would like to refer anyone seriously interested in improvement to Alistair Cockburn's Oath of Non-Allegiance. It reads as follows:

"I promise not to exclude from consideration any idea based on its source, but to consider ideas across schools and heritages in order to find the ones that best suit the current situation."

We are learning the wrong lessons from our politicians. We need to stop demonizing those with other perspectives, become a little less ideological, and learn to work together.

39. Pingback: 《*Scrum*要素》参考资料汇总 | *Scrum*要素

40. Pingback: *Kanban vs. Scrum: Kanban is NOT for Software Development, but Scrum is! « The Scrum Crazy Blog*

41. Y8 Dress Up *says:*
April 29, 2013 at 9:32 am
Reply
Its such as you learn my mind! You appear to grasp a lot approximately this, such as you wrote the guide in it or something.
I think that you simply can do with some p.c. to power the message home a bit,
but other than that, this is great blog. A great read. I'll definitely be back.