Table of Contents

Blog Entries

- Do We Need Points To Generate a Release Burn-up Chart?
- Fixing Defects Does Not Mean You Are Addressing Technical Debt
- How Can We Determine How Many Undiscovered Defects We Have?
- How Can We Improve Collaboration on User Stories?
- How Can We Understand the Real Value of Fast Feedback and Deciding Late?
- How Can We Work More Effectively with Remote People?
- How Do I Lead, Practically?
- How Do We Allow for Innovation?
- How Do We Deal With Product Backlog Explosion?
- How Do We Improve Our Improvement Process?
- How Do We Improve Team Performance?
- How is Work Assigned During a Sprint?
- How to Preempt Team Conflict?
- It's Not Just The Meeting
- Our Estimates are Terrible!
- Presented at Agile 2016
- The End of Enterprise IT
- What Is The Effect of Changing Team Members on Velocity?
- Why a Plan Based on Average Velocity Will Fail?
- Why Doesn't Traditional Project Management Work For Software Projects?
- Why Is Agile So Hard - The Backward Bicycle?
- Why Should We Work Harder to Eliminate the Effect of Dependencies?

See Blog Entry Tags to understand subjects.

Last 5 Blog Posts

How Do We Improve Our Improvement Process?

Work in progress

One thing we talk about with agile is the idea of continuous or relentless improvement. Most organizations want to improve, but like a lot of things it is sometimes hard to feel good about the improvements you are making. Further it is hard to establish a continuous practice where the whole organization is doing these improvements and feeling good about it.

One tool that I've seen helps organizations is to treat improvements as experiments and talk about them this way. The traditional approach to capturing improvement is to set up SMART goals. The problem I have with this approach is that we are treating the improvement as a conclusion (it will result in the improvement we expect - its deterministic) rather than recognizing that the new approach might in fact result in no improvement. And when this happens the thinking is that you have “failed” because you did not meet your goal.

Learning is not failure. And to capture this instead of talking about a change or an improvement we set up an experiment.

To define a good experiment we need a definition of what the experiment is, what the expected result is, what the result actually turned out to be, what we learned, and what we will do as a result. This is the scientific method.

And I would then maintain a list of experiments we have run somewhere in order allow others to learn from our experience.

So, for example, during a transformation to agile, we might want to understand whether the idea of a “True Team” (see Why Do We Form Teams When We Transition To Agile?) actually is a concept that works in our organization. We could setup and experiment as follows:

- Experiment: Does a “stable team” result in increased engagement (with the work and with other team members) when team members are in multiple locations but in compatible time-zones?
- Label / category: Stable Team
- Expectation: Team members will report improved engagement from team over their traditional working method as a result of moving to a stable team structure. Concern is that “remoteness” will cause a reduction in team engagement.
- Metric: Survey questions (note: example provided for discussion only – expect will need some better design):
    - Question: “In comparison to how you have worked in the past, how would you describe you level of engagement with other people working on the team?” – “We are not working well together”, “We don’t seem to be working together as well as we have in the past”, “I’ve seen no change in how we work together”, “We seem to be working a little better together than we did in the past”, “We seem to be working together much better than we have in the past”. Plus a comment field “If possible, please provide a specific example that supports your view.”
    - Question: “In comparison to how you have worked in the past, how would you describe your understanding of the goal of the work?” … or perhaps this is a separate experiment …
- Timing: Do survey 4 (say – need to set up in time for rollout) weeks after team kick off, analyze results and provide feedback one week after that
- What Did We Learn?: TBD

Then, when the time box is finished, we collect the result, and determine what we learned and what we do next. (Hopefully in this case you got the expected, positive result:-))

Experiments can be done at all levels. Teams can set them up as part of a retrospective. Transformation coalitions can set them up as a result of their retrospectives. If you are doing SAFe experiments would be generated as part of a the Inspect and Adapt. And so on.

Using this approach at whatever level you operate at helps you become a “learning organization” where experiments are seen as a normal part of the work you do. The vocabulary will help and assists by making change less threatening (more “try it and see” and less “you will do it this way”). In particular we want to treat new understanding as a “success” (and publish it as a “win”) to aid the momentum. It also like to encourages a spirit of experimentation beyond just “lets try it!” (said in giggly tone). Experiments should be aimed at generating useful knowledge.

2017/03/21 09:49 · Hans Samios · 0 Comments · 0 Linkbacks
blogentry, faq, todo, experiment, improvement

The End of Enterprise IT

I came across this article by Mary Poppendieck recently called "The End of Enterprise IT". Mary is truly a thought leader - I find that she is able to cover topics that I am thinking about with tremendous transparency and focus. Mary tells the story of an agile transformation at ING Bank. ING Bank went to Agile in their IT organization and while they saw benefits, they did not feel like they were getting the transformative results they really needed. In particular, they felt that while their business people were “involved” with the IT organization they were not really aligned in the mission.

The leadership of ING Bank looked at their organization and came to a startling conclusion. We all know that according to Marc Andreessen "software is eating the world". From Mary, ING Bank took this to the next logical step:

“The leadership team at ING Netherlands had examined its business model and come to an interesting conclusion: their bank was no longer a financial services company, it was a technology company in the financial services business.”

What this meant was that in order to compete, in order to be the bank they wanted to be, they needed to set themselves up more like a technology company where good engineering is a necessity. “Engineering is about using technology to solve tough problems; problems like how can we process a mortgage with a minimum of hassle for customers? How can we reduce the cost of currency exchange and still make a profit?”

They found that technology companies did not have Enterprise IT organizations. And “even though they were much bigger than any bank” they did not “have much of a hierarchical structure.” “Instead, they were organized in teams – or squads – that had a common purpose, worked closely with customers, and decided for themselves how they would accomplish their purpose.” Business, engineers, everyone, all aligned.

As a result, ING Bank “chose to adopt an organizational structure in which small teams – ING calls them squads – accept end-to-end responsibility for a consumer-focused mission. Squads are expected to make their own decisions based on a shared purpose, the insight of their members, and rapid feedback from their work. Squads are grouped into tribes of perhaps 150 people that share a value stream (e.g. mortgages), and within each tribe, chapter leads provide functional leadership. Along with the new organizational structure, ING’s leadership team worked to create a culture that values technical excellence, experimentation, and customer-centricity”

When I work transformations with organizations, too often we see it applied just to the IT shop. Now don't

get me wrong, things usually improve when you do this both in terms of the business results and the way people work. But the issues that then remain are often where the business intersects with the IT shop. To me, ING Bank thinking represents a way to address this. More importantly if you are in a business where a lot of what you do is driven by computers or programs, then perhaps you need to look a little closer at your business and determine if your thinking needs to shift to be “a technology company in the X (e.g. financial services, insurance, etc) business.”

Want To Know More

- "The End of Enterprise IT" - Original article from Mary Poppendieck
- https://www.youtube.com/watch?v=6FPXbQ2WpAM - Video of ING Bank's journey. Message is that starting in 2008, ING decided that all their services are being provided as software and that, in fact they are more a software / technology company than anything else. They moved from a traditional waterfall / mainframe / project management approach and moved to agile / devops / continuous delivery of value mode. They have 180 teams developing software, and now do 1500 deployments / week to production (I think, at least that was the implication).
- "ING's Agile Transformation" - An interview with ING Bank's CIO Peter Jacobs and (ex-COO) Bart Schlatmann
- "Building a Cutting-Edge Banking IT Function" - An interview with Ron van Kemenade, the CIO of ING Bank

2017/01/17 08:03 · Hans Samios · 0 Comments · 0 Linkbacks
enterprise, blogentry, transformation

Why Is Agile So Hard - The Backward Bicycle?

Premise

One of my favorite quotes about an agile transformation is “If you think Scrum (/ Agile / SAFe) is easy, just try it”. I have no idea who said it first, but it captures a lot. Done right, the move to agile will make visible all the problems you currently have, and then gives you a couple of weeks to make progress on them. While this is all going on there are subtle shifts in the thinking process that you have, which adds to the confusion (for more see What Are The Changes in Culture That Need To Happen with Agile?)

But under all this, I think there is something more subtle going on and that is the difference between “knowing” something and truly understanding that thing. A colleague recently sent me a video that really helped me understand the difference:

- The Backwards Bicycle / Brain - https://www.youtube.com/watch?v=MFzDaBzBlL0

I think there are a number of other interesting lessons from this video:

1. Clearly “knowledge” is not the same as “understanding.”
2. You will often not understand something until you have done it yourself. Can you say “gemba?” It is especially interesting to me that people's reaction to the backward bike is to scoff at the person

that is having such a hard time because it is clearly so easy to do. “In theory, theory and practice are the same thing. In practice …”
3. You have biases, and you are probably unaware of them.
4. If you already “know” something, learning a variation on what you know is especially hard.
5. If you are older it will be even harder to unlearn something. Which is also related to the idea that if you “know” something for a long time, it will take a long time to undo that learning

Since most of us have been working software development for a lot of years, is it any surprise that we have difficulty changing how we think about the problem?

This is what I got out of the video - what else did you learn?

Additional Learning

I've been an agile coach for a while and have used the video of the backward bicycle a lot to help people understand the mind shift required. Intellectually I understood the lesson. But one day we actually got one of these bikes and we were able to try it.

And what I learned really surprised me. What I learned was that while I intellectually understood the message I really did not think it applied to me. In my heart of hearts I expected to actually be able to ride the bike. When I climbed on, it was amazing to me how I reacted - I just am amazed that I am unable to do anything with the bike.

To emphasize, knowledge really does not equal understanding.

2016/10/26 06:16 · Hans Samios · 0 Comments · 2 Linkbacks
blogentry, knowledge, understanding, faq

How Can We Improve Collaboration on User Stories?

> “Everything is vague to a degree you do not realise till you have tried to make it precise” – Bertrand Russell

When it comes to understanding and documenting requirements there always seems to be a discussion and room for improvement. In general we capture a user story in an automated tool with a summary / title and a description that has both the user story (“as a”, “I want”, “so that”) and a section describing the “Conditions of Satisfaction” (aka “Acceptance Criteria”). People wonder “how much is too much detail?” Others wonder whether they really understand the requirement in enough detail to do something useful. Still others think that we should write a complete specification.

One team I worked with recently ran an experiment using a slightly more formalized approach to capturing Conditions of Satisfaction which helped us understand the requirement, but also did not specify how solution was to be derived. At the time we were trying to solve a problem with the (acceptance) testing of the workflows but found that the solution we worked also helped us improve the discussion and collaboration around the requirements.

By way of background, when we tested during the first phase of the project, while we had good requirements, we really had not taken the time to develop a good test plan, relying instead on an ad-hoc approach. The result was that we did not feel we had covered all the basics. We felt that we had duplicated efforts in some areas. To improve we decided to capture the acceptance tests during the development of the requirements.

For the second phase of the project we used a format for the Conditions of Satisfaction which is related to acceptance test or behavior driven development. The basic format of the acceptance criteria is described as:

> Given <some initial context>
> When <an event occurs>
> Then <ensure some outcomes>

The work we were doing involved the flow of data between Siebel and JIRA. Let's say we have a user story "As a support engineer I would like see what is happening as people discuss and work a Siebel issue so I can ensure that it's heading in the right direction", we would then write the Conditions of Satisfaction as "Given a person is in working in JIRA Agile, when the person adds a comment to the JIRA issue that is linked to a Siebel CR then the related Siebel record notes will by updated with the comment".

Just like the user story format helps us capture and understand not only the feature but who needs it and why this format for the acceptance test helped us capture and understand the acceptance criteria from an end user's perspective. It also helped us understand what it meant to test the capability when that time came around, forming the basis of our test plan. It really increased the amount of discussion we had about the requirements, and made things clearer for everyone.

Some of our requirements did not easily fit into this format. For example we had a number of requirements that given a condition, we'd have certain mappings go between the two systems. For these we simply captured a table of conditions, inputs and outputs. I will say it sometimes seemed that it took a while to get to a definition that we all agreed on, but I think that is the point - this also helped with collaboration and clarity as well.

It should be noted that when you capture acceptance criteria in this format they can also be the basis of an automated test using tools like Fitness or Cucumber to actually run the acceptance test. But even if you do not do this, it is worthwhile experimenting with the approach as it really helps with collaborative communication ensuring everyone gets on to the same page.

Note: Good starting book to read if you want to learn more is "Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing" - Gojko Adzic.

Note: this posting was originally published in August 2015 on a Wordpress Blog

2016/10/12 13:01 · 0 Comments · 0 Linkbacks
userstory, blogentry, faq, acceptancecriteria, conditionsofsatisfaction

Presented at Agile 2016

I finally got around to publishing the presentation I did at the Agile 2016 Conference this year. Title for my session was "How to Work Personality Issues Without Sounding Like a Marriage Guidance Counsellor?" It was basically a presentation on approaches you can take to deal with conflict without coming across as disingenuous.

The session was well attended (about 120 people) and received positive reviews, so I was happy. A number of people came up to me during the conference to talk to me about the specifics.

Full set of notes and materials are at Hans Samios - How to Work Personality Issues Without Sounding Like a Marriage Guidance Counsellor? The PowerPoint slides includes a full set of notes.

Also, my notes on the conference are at Agile 2016 Conference in Atlanta, GA. Note that this is still a bit of a work-in-progress.

Read and enjoy!

2016/08/15 18:27 · Hans Samios · 0 Comments · 0 Linkbacks
blogentry, conference, agile2016

Older entries >>