Agile Thinkers

A blog about the many great thinkers in the Agile community. Hosted by Clarke Ching

June 08, 2008

Tom Gilb - an example from Ryan Shriver

I'm not sure what you call people who follow Tom Gilb's work ... Gilbettes, perhpaps? Gilbittes? Gilbees? Who knows?

Anyway, I'm one and so is Ryan Shriver. Here's Ryan's stunning story of how he enhanced his already impressive Agile work by following Tom's advice.

Q: My first question: is your story the same as Jens'?

No, not really. I'm envious of Jens' story. My implementations of Evo + Scrum haven't been as clear-cut and clean.

I got introduced to Tom and Evo in Sept 2005 when I presented Scaling Agility at the Agile Business Conference in London. My talk was on how you do XP on teams of up to 100 for product development. It contained lots of "tricks of the trade" and "lessons learned". I got good feedback from people in the field struggling to apply these concepts on their projects.

As background, I learned Agile (well, XP) by reading Kent Beck's white book and putting pieces into practice on projects I was working on. This was around 2000 or 2001. Our team was floundering and as a senior developer I was seeking out better ways to work together when I came upon XP. We implemented some of the development practices (like unit tests, pair programming, etc.) but it wasn't a whole-team adoption.

As I moved onto newer projects, I'd do a little more XP each time. In December 2003 I was brought in as the lead architect for a product company that needed to get a new product to market (large loan system, J2EE, relational databases, rules engine, web based, etc.). I convinced the COO to use XP for development and she agreed (I learned later that prior to my involvement, her top analysts had spent 9 months trying to identify requirements and design and had gotten essentially no-where. That, and they had a signed contract and needed to show something soon, helped convince her to try XP).

From Dec 2003 until Sep 2005 the team did XP and we grew from 6 developers / 12 persons to 40 developers / 100 persons on the project. I was both the technical architect and agile coach. We had some successes, but also struggled with aspects as we got bigger that bothered me. I gave the Scaling Agility (slides are on my site) about this time that talked about the positives and negatives we experienced. Some of the issues I struggled with included:

1. Thousands of user stories with no clear sense of what was REALLY important to the stakeholders. The Customer Team wrote lots of stories but there was no higher-level sense of the larger objectives for the system.

2. "Non-Functional" requirements were practically non-existent, even when I asked I got vague answers. The product company typically relied on their customers to provide these. We'd design to handle one customer's volume, only to have a new customer come along with 10X the volume and we spent too much time redesigning and refactoring working code to handle the higher volumes. This was doable with agile, but very expensive and time consuming. As an example, one large customer had 4,400 functional requirements in their RFP for the system and less than 40 non functional requirements. I'm not kidding. This company was going to put about $60 Billion USD on this system. Both parties severely underestimated the system qualities and were very far apart in what their expectations were. Agile didn't have much to help here, aside from "create user stories".

3. Overall frustration that while XP worked well at the development level, there was no way to report "progress" aside from what stories got done. And that doesn't mean anything to powerful people who make important decisions :-) We could only report 'velocity' based on done stories and this didn't make sense for the COO and other senior executives. I had no way to tell them how we were delivering value, if we were at all. No idea the business value of some stories over others.

So I present at the conference and hear Tom for the first time. He was doing a keynote and he said something like, "Unless agile methods are quantifying critical success criteria, they're incrementally better than waterfall methods". I was floored. In a good way. At an agile conference full of agile cheerleaders all drinking the agile kool-aid, this guy I've never heard of was speaking some truth. And some of what he was talking about seemed to address my needs and concerns using agile and my complaints about it.

So I tracked him down in the hallway, we found a table and Tom starts showing me impact estimation tables, the concepts around objectives, targets, failure and parts of Evo. I picked up his copy of Competitive Engineering at this conference and started reading it on the flight home. It was a dense read, and it would take me a few readings to get through a chapter, but I kept at it for the next 4-6 months. I'd read a little, apply some of his concepts to my project, and figure out what worked, what didn't and how to fit the concepts to my team.

Because I was the lead architect, I first put Tom's principles into use on the performance testing and non-functional requirements for a major client of the software vendor. These were areas I was working on and tuning system response times, transactional throughputs, job run times and other things like them benefited from impact estimation tables. Once the performance objectives were defined with target and failure level, we used application profiling to identify the bottlenecks and to estimate the time savings we'd achieve if these bottlenecks were removed. Once we'd get an estimate for the potential savings, we'd estimate the complexity to make the change in the system (these were often cross-cutting changes that had to be thoroughly tested and thought-through). Using these, we'd identify the "biggest bang for the buck" change, make it, and re-test performance. We kept doing this until we got performance in line.

Oftentimes, our "gut" about which changes to make next were the same as the IE tables indicated, but I always felt better about having numbers to back-up my decisions about which changes we'd do and why (performance to cost benefit). I found that some of the other architects liked these methods, some didn't see the purpose and others couldn't care less...it didn't change their world. So adoption on the team was a mixed bag.

Our performance turning resulted in two high-profile benchmarking sessions at Sun Microsystems for an important customer. There was lots of fancy expensive hardware and the system performed well. It was stressful for me and our group of engineers with the client's team in the room while we dialed in the software and hardware

I left this project in Feb 2007 and started doing two new projects, in an agile coaching and requirements analyst capacity. Around this time I switched from XP to Scrum, mostly because Scrum was being better marketed here in the US and it was becoming the "corporate agile" method of choice. I actually like some aspects of Scrum better, such as the Product Owner vs. Customer Team, but even as I switched to Scrum I found out that it had the same shortcomings as XP with respect to issues I mentioned earlier.

I kept reading Tom's book and looking for ways to apply the concepts I'd used to performance test the system to cover some of the shortcomings with XP (and now Scrum). When I started the two new projects, I had some new opportunities to try out some of Evo concepts (for planning and requirements) with Scrum (for delivery).

The first project was for a local start-up in the travel industry. It was a subsidiary of a global company. The were / are building a patent-pending product for cross-selling products and services to people who travel. We pitched the CEO in the proposal to do the project agile with measurable business goals, and our other three competitors all pitched waterfall. We won the work :-) I did the inception phase almost solo, documenting the high-level business objectives and system quality objectives. I used Tom's planguage to do this. Also did the high-level functional and quality (performance) requirements along with a high-level architecture of the solution.

I found the CEO liked the ideas of the objectives, but didn't embrace them like I thought he might. The director of development, who I worked closest with, loved them though and continues to use them. The CEO agreed to 13 high-level business objectives that I defined, developed target and failure levels for and said we'd use to prioritize our product backlog. I spent a good bit of time getting these done, but as I transitioned to a larger team from our company who came in to deliver the system in the next phase, this work really helped set the project up for success.

We ended working with them for another 12 months delivering newer versions of the system. We'd re-evaluate the progress towards goals each quarter, make adjustments, and continue. The team was doing eight-week releases and two-week sprints so about every 1.5 releases we'd revisit the goals (should have been every release). We're working with this client right now to produce a case study of this project, but unfortunately they haven't been as diligent about running down actual results and sticking to it as I'd like. I think we will get some numbers that we can share, but it's not as perfect of a case study as I'd like.

But the system has been a big success, launched with 4 major airlines and travel companies. They're doing about 1 million daily web-service transactions and volume is growing as they add new partners. One of the things that I really focused on when I was doing requirements is the response time and throughput transaction performance. I set aggressive target and failure levels for each that would allow them to grow the business without major re-designs for performance. Target response time was < 500ms with failure > 2500ms. Target throughput was 50 TPS if I recall, more than enough for the to grow the business. From day 1, the architects, developers and testers knew these numbers so they designed to hit them and early on we put in load testing and monitoring. They'd test, I'd help them tune the system, repeat and continue. To this day they've had near zero issues with performance even there volumes are growing rapidly. I attribute this directly to the clarity that planguage brought to the problem and the teams focus on a few performance metrics that were important.

The second project was for a Fortune 500 financial services company (all these are in the USA). They had an IT department that looked at new tools and methods and wanted to learn agile. The found us and we agreed to help teach them agile. But the one thing that differentiated us from others is we emphasized using agile to deliver measurable business value with Scrum. We won the work and I had to get on a plane! The team didn't know how to select the best project to do Agile with, so I helped them use Evo to figure this out. We found one of the lines of business as a willing partner, did some interviews with people at all levels (associate to CEO of a line of business) to understand better they're challenges, frustrations and get a sense of where we could focus. We also looked at the organization's objectives to see what they said. We did a workshop with the entire team (executive sponsors, sales, technologists, analysts) and validated the top 4 business objectives and zeroed in on "Increase Revenue" which was the most important. We asked the sales team, "what could we do to help increase revenue?" and they said, "Give us more time to sell. We've got lots of issues with our systems, but if we just had more time to sell we'd generated more revenue".

We started identifying that the sales people did a lot of their own administrative work when not selling and they struggled to find out what forms to use to open new accounts. Some of them also had labor-intensive processes for creating new letters. We identified 13 possible opportunities out of this session that could help us increase revenue.

Then the technologists were provided the constraints of the project: Six weeks, small development team. That's it. We asked them which of the opportunities could even be impacted in that time. They said it had to be established technology and involve next to no system integration. So that ruled out some options and eventually we can back with three options to the business, any of which could make some improvement towards in six weeks.

We collectively agreed to focus on the system objective "reduce the time it takes to open a new account". We used impact estimation and zeroed in on the design that asks some simple questions (between 2 and 7) and then returns the right account form based on configured business rules. In six weeks the team designed and implemented the three-screen interface and got 24 new account type rules configured in the system. A few other minor features but that was it. It was a big success. Small, simple and was done with minimal costs. Now it's rolled out to about 100 people in the field. That was last year and this year I'm following up to get results, but it's been difficult to get more than qualitative success.

So I wish I had more case-study data to back these up, and I'm working on each, but I don't have as much as I'd like. There is a third project I coached early this year that went from measurable objectives to backlog. This project has the best success because the company really bought in. Release 1 just rolled out and the goal is that by July we have our first feedback numbers. I'm seeking permission to publish it or at least share it at Agile 2008.

So that's my story ;-)

Oh, your questions:

Q: Tom said that you've been using Scrum combined with EVO. Can you tell me a little background about the project and what problem EVO solved that Scrum didn't?

See above!

Q: How did you learn enough about EVO to get started? Was that easy?

Met Tom, bought Competitive Engineering. Taught self, with emails and reading papers from Tom. Also attended his seminar last year and plan to this year. Tom's very giving with information and his time and this has helped quite a bit.

Q: Did you have to change Scrum much?

You don't really "change" scrum, you just augment it with Evo. You change your thinking about how you prioritize your Scrum backlog, but the mechanics of Scrum still work the same (stories, sprints, retrospectives, stand-ups, backlogs, etc.). Essentially, the parts of Evo I adapt don't really have a counterpart in Scrum, and while there is some overlap between the two (Evo has Implementation Cycle, Scrum has Sprint - same concept), both methods are also complimentary if the best of each are used together.

Posted at 10:26 AM in Tom Gilb | Permalink

TrackBack

TrackBack URL for this entry:
http://www.typepad.com/t/trackback/12757/29935452
Listed below are links to weblogs that reference Tom Gilb - an example from Ryan Shriver:

Comments