13 OCT 2010

Why Value Points? Using Agile to Optimize Value of Projects

https://www.thoughtworks.com/insights/blog/why-value-points-using-agile-optimize-value-projects

Elena Yatzeck

Pragmatic Agilist

TransformationStrategy

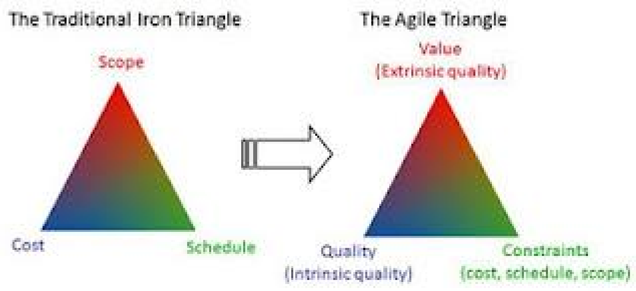
I was asked recently for guidelines on "how to use value points in agile projects". I was glad to get the question, since some people, like the blogger who writes "Agile 101," say stuff like: value points are "not appropriate or particularly necessary in all cases." Gah! The Agile 101 author actually goes on to do quite a nice job describing how to use value points, and I recommend you visit the blog, but I would like to explain why I don't think all of this is an optional nicety.

If you were CEO of your company and you were being briefed on your project every two weeks, what do you want to hear from your agile team?

1. "We delivered 14 value points this iteration" (which translates into some specific thing the company values, in terms of cost avoidance, expected revenue within a portfolio, increased quality, or faster time to market)
2. "We delivered 14 story points this iteration" (which translates directly into "2 weeks of effort")

I think the answer is pretty clear.

Jim Highsmith first published details of this proposition in this blog entry on the Cutter Consortium site, and expands on it in the second edition of his watershed book, Agile Project Management. In his book, he also expands on this concept of the "Agile Triangle" of Value, Quality, and Constraints, which replaces the traditional Project Management Triangle of Cost, Schedule, and Scope.



Highsmith's proposition is that when your company embarks on a project, you are making a financial investment in order to produce something of value to the company. That value has two dimensions:

- Value (extrinsic quality), which must be determined by the people in your company who are going to use the new software to create revenue for your company. They will sell the software, or sell the things the software help them to build, and they should have an idea from market analysis and company strategic goals why they want to spend money on the project at all, and how much money the project is worth to company owners (stockholders, public bodies, or a private individual).
- Quality (intrinsic quality), which must be determined by your software architects. They will maintain the software you are building over time, and their goal should be to minimize the cost of this maintenance over time.

Product managers can predict the revenue your company will gain from completing each feature of the project, and architects can predict the revenue your company will lose over the projected lifespan of the project by cutting corners in design at the outset. "Cut corners" in design are what Highsmith and others call "technical debt." Bad software is a hidden cost waiting to hit your company in the future, when you suddenly want to do something new and find you can't do it without starting all over, because what you built isn't flexible.

So as a company invests in a project, it should maximize the immediate and long term value from that investment, while operating within the constraints of cost, time, and potential features to be included.

https://www.thoughtworks.com/insights/blog/why-value-points-using-agile-optimize-value-projects

Okay, you say, I feel motivated today. I want to maximize my project's value, not just track the cost of the effort to deliver it at full scope. How does that work?

Here are some mechanics:

- For each project, assign an IT Lead to monitor what Jim Highsmith and others call "technical debt," so that as the software is developed, expected costs of maintenance do not cut into the financial values being tracked by the Product Owner.
- For each project, assign a Product Owner, a decision maker whose job it is to predict, record, and then *update the business value parameters for the project*. These consist of:

    - **Project value:** the overall value of a planned software project to your company (predict at project inception, and update before each release planning meeting as needed). This can be a financial return on investment, or some other appropriately quantified measure, such as increased speed to market, increased quality, or avoidance of future costs. Cost avoidance could include such factors as regulatory compliance to avoid fines or building software to allow something new to be done in an automated way, rather than taking on new staff.
    - **Feature value**: the relative value of each requested software feature which is planned as part of the overall project (predict during project inception and update before each release planning meeting as needed). Feature value should be determined through a method such as planning poker, which

allocates value to specific features on a relative basis. So the Product Owner should gather the right people together to think about the expected high level features of the project, and assign a "points" value to each feature. Once a sum of points is established, that can be mapped to the overall predicted financial value of the project as a whole, and each value point is worth some fraction of that overall value.

    - **Story value**: the amount of feature value delivered by each story, where each feature is divided into stories which can be delivered in 1-5 days. The rule is that the story must provide some recognizable business value on its own. (Assign as stories are carved out of features, during release and before or during sprint planning). After the team determines which features should be targeted for a release, the first features to be delivered within a release should be divided into stories which deliver some proportion of the feature's business value in their own right, end to end. Stories should be things which can be completed within a sprint.

You can readily see that once you are tracking value at project, feature, and story level, you can do a whole bunch of excellent things, *each of which optimizes your company's bottom line*:

- Plan your portfolio: now that you know the overall expected return on investment of a project, you can weigh that project's risks and potential returns and determine when to make it part of your active portfolio.
- Speed up and maximize the return on investment for all of your projects: each project is executed as a series of releases which have

the most valuable features in the earliest release, and the most valuable work delivered earliest within each release.

- Graphically track the value delivered by the project during each sprint and for whole releases, using a "value burn-up," (exactly like an "effort burn-up," but measuring something a lot more interesting to your CEO)
- STOP your project once you get decreasing returns on investment, in favor of starting a different project in the portfolio which has higher-value features still to be delivered.
- Collect returns on the working software through the expected lifespan of the design and beyond.

One final point. Agile software development is premised on avoiding "Big Upfront Design" (BUFD), and so projects are started with some minimal inception phase to lay out an initial architecture design and an initial effort estimate. All designs and estimates are expected to change over the course of the project as more is learned.

Value points should not take years to estimate any more than your effort points do, even though they result from different techniques. Do not leap out of the frying pan of the BUFD into the fire of Big Upfront Value Modeling (BUVM, sadly not pronouncable).

Your decision-making Product Owner should try hard to limit the time she spends estimating its value. Just as you shouldn't "start coding" on day 1 of an agile project, you shouldn't "pick a number out of a hat" as your business case. But neither should you over-analyze, since you expect to be enhancing and revising your business case as the project proceeds. And of course the market is going to be changing during that time anyway! So as the project unfolds, the Product Owner should plan to revise the overall predicted project value and value of desired features on a regular basis, and the whole team should plan to adjust release plans appropriately as they go.

The main point is that value points, like their better known cousins, effort (or "story" points), can and should be estimated, written down, adjusted, and used for planning and reporting purposes by project teams. They can be estimated in Planning Poker, burned up, burned down, and written in the upper-left-hand corner of your story cards. ONLY value points, however, can tell you how well the project is returning on investment. And that's why I don't think they're ever optional.

This post is from Pragmatic Agilist by Elena Yatzeck. Click here to see the original post in full.