david consulting group
TRUSTED ADVISOR

Is Calculating ROI Meaningful for Agile Projects?

December 2014

Scope of this Report

This report is not about ROI of agile methods versus other SDLC's. Instead, we consider if the traditional approach to producing business cases for projects or programs by predicting financial outflows (project costs) and financial inflows (new income of savings) is still appropriate or even meaningful for agile software development based on scrum and/or enterprise wide extensions of scrum such as SAFe or DSDM.

Definition of Return on Investment (ROI)

Description	Ratio of the benefit a project yields to the cost of the project.
Purpose	Evaluate between features, releases or projects based on ROI and on overall performance after implementation.
Utilization	As a planning tool to evaluate whether to include features in a release or projects in a portfolio.
Data Required	Cost, estimated benefits (expressed as a stream)
Calculation	Simple: ROI = ∑ (Net Present Value of Benefits)/ ∑ (Net Present Value of Costs)
Timing	Planning
Baseline	Company Hurdle Rate
Industry Data	Not Applicable

The Traditional Approach to ROI

The traditional approach to justifying software development investment has been to draw up a time line of predictions of project costs and project income or savings in the form of displaced current costs. The "go – no go" decision is then based upon company targets for one or more of simple ROI, Cash-on-cash

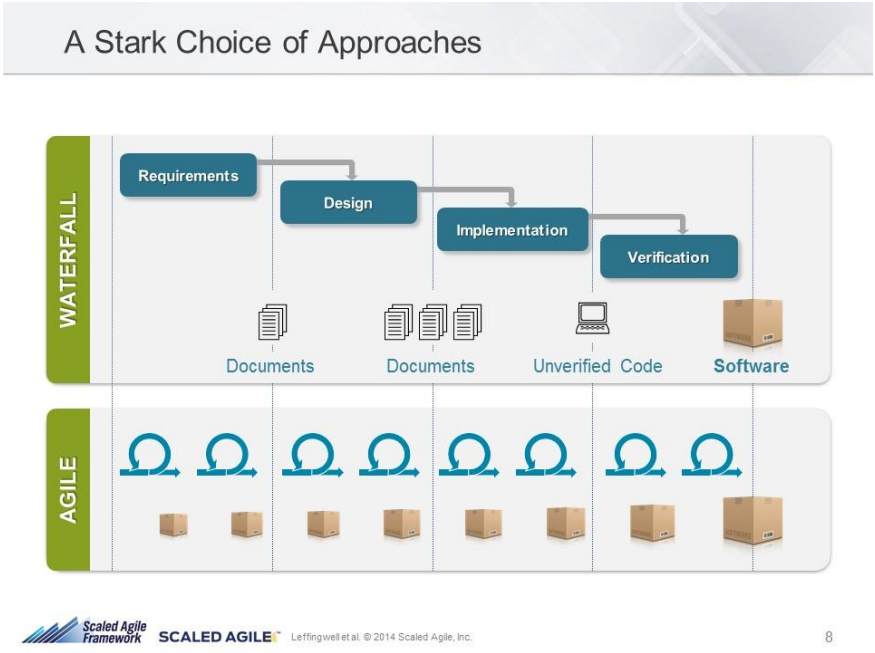
return, payback period, cost of money or some other similar measure. ROI has always been useful for comparing alternative projects or investments when funds are limited, as they almost always are!

Appropriate Granularity for ROI

Traditionally, we have worked with software development projects which are sometimes aggregated up into programs. Of course, there is no universally agreed definition for these terms and different organizations have had different financial cut-off points above which investments in projects/programs require business cases and for which those business cases must return a minimum ROI to proceed. Once a commitment has been made to fund a project /program based on a business case, there is an expectation (however naïve) from stakeholders that all of the elements of the software needed to deliver the estimated financial returns will be completed with a manageable increase in estimated costs. Any other outcome threatens the viability of the original funding business case. Perhaps it is for this reason that few organizations track the actual cash flows from projects very carefully. Or perhaps it is simply that the asynchronous nature of the cash flows - "out" during the project but only "in" when the software is employed by the business – is beyond the attention span of both IT and the business. This is a gap that Finance could fill.

In contrast, for Agile software development where most organizations start with Scrum, the future workload is represented by a backlog of epics that break down into user stories which form the incremental deliverables. Stories, by design, are much smaller units of work than projects (see Figure 1):

Figure 1: Contrasting models of value delivery (Source: Scaled Agile Inc.)



In Tara Hamilton-Walker's blog post in 2009 on "Self-funding Projects", she notes, "In reality, it is unlikely that you would/should take the time out to calculate the ROI for each story." Indeed, the

incremental delivery of value raises some important questions that go to the heart of the title of this report:

- Is it possible to associate future income/savings with a single story? We agree with Hamilton-Walker – while it may be possible it is almost certainly not practical.
- Is it worth investing in the cost of building a business case for an individual story? Again, no.
- Can we conclude that it is not meaningful to calculate ROI for Agile Projects? Not yet – but we do have to get away from the concept of "Agile Projects" and consider further the appropriate level of granularity for ROI for Agile Software development at the epic or functions (which might be groups of epics) level which are business recognizable.

One approach to calculating ROI for Agile

In her blog post, Hamilton-Walker goes on to assert that one of the biggest benefits of developing software in an iterative or agile way,

> *"… is that you can start realising the benefit of your work before the project is officially 'finished'. The objective is to front-load the project with the most profitable or highest value deliverables and release them as soon as possible so that they can start making money for you. If you're using a Lean approach, you'll release them as soon as they're finished, if you're using a Scrum approach, you'll cluster a few into one (or more) releases at the end of each iteration."*

This concept of delivering value early and continuously is one of the foundations of Agile (See Figure 2) and the breaking down of enterprise level demand (and cost) into lower level backlogs then ensuring that the resulting value (income or savings) is a critical benefit of the Scaled Agile Framework® (SAFe).

Figure 2 Value Flow from Agile (Source: Scaled Agile, Inc.)

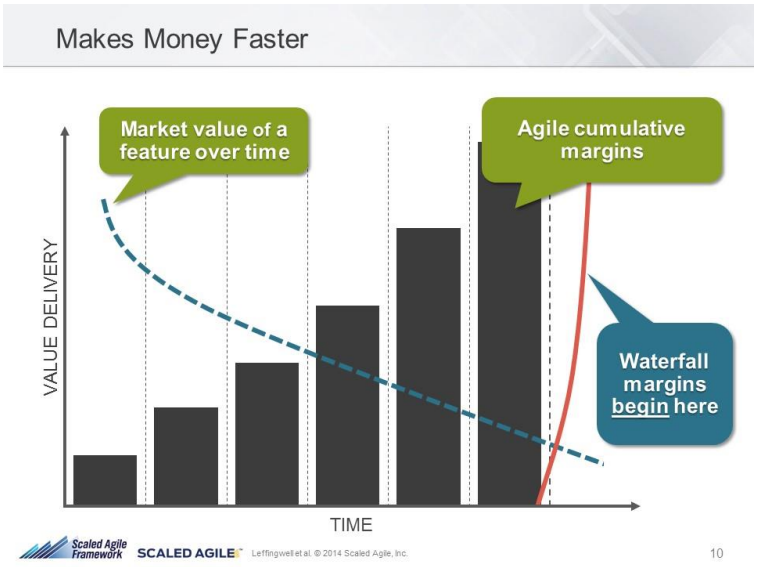


Figure 2 also illustrates another benefit of Agile from a ROI perspective by identifying the diminishing market value of a feature over time. This effect is separate from and additional to the use of Net Present Value (NPV) in ROI calculations.

In his blog post, Steps for Calculating ROI on Agile Projects, Richard Banks identifies a key challenge with calculating ROI for Agile at the higher levels of granularity, the final scope of the deliverable at any given future time point is not fixed. As he states, "In an agile project requirements are often discovered, adapted and/or changed completely as the project progresses and very little up-front design work is performed as a result, unlike traditional methods that work on the belief that requirements can be defined up front and then fixed for the duration of the project."

Banks proposes using the Product Backlog as the appropriate aggregation of stories for ROI. He suggests an initial ROI calculation can be done broadly as follows (with our proposed modifications included):

1. Gather the initial high level requirements for the project (presumably the project is describe as a theme or one or more epics) and place them in the product backlog
2. Get the team to estimate sizes for the requirements individually at a high level using story points, or whatever unit of measure they are comfortable with, but keep it high level and estimate quickly.
3. Examine the overall size of the product backlog. If the size seems too large at the start, then it probably is. Before proceeding either look to cull items from the backlog, or cancel the project.
4. Work out your starting velocity (i.e. how many points your team completes in a sprint). If possible, get approval to develop one item (or a few) as a proof of concept exercise - this often helps determine what the initial velocity of the project is likely to be.
5. Extrapolate from your initial velocity how many sprints will be required, and thus what the cost will be.
6. Find out from the business the following income or savings metrics:
    a. One or more units of measure in which the income or saving will be denominated e.g. "new subscribers added" for a website
    b. The number of units of value associated with delivering all, or subsets of, the Product Backlog e.g. this story/group of stories/epic will deliver 100 new subscribers
    c. The actual monetary value of delivering a unit of value at particular dates on the project timeline and beyond e.g. Each 10 new subscribers are worth $10 on March 1, $15 on April 1, $20 on May 1, $15 on June 1 and so on.
    d. The cost of delay, if any, of delivering all, or subsets of, the Product Backlog e.g. this story/group of stories/epic will deliver cause us to pay a fine of $2000 if we do not deliver by August 1 or this story/group of stories/epic will deliver cause us to pay a fine of $2000 if we do not deliver by August 1 be worth nothing if delivered after the Christmas sales season.
7. Use the estimated time, income, savings and cost figures to determine your ROI.

Of course, the product backlog, income and saving metrics and velocity can and will change and that must trigger recalculation of the ROI. When? It is appropriate to calculate the sensitivity of the business case to changes in the underlying metrics and set up allowable ranges of fluctuation for individual metrics, beyond which the ROI must be recalculated. It is certainly not Agile to continue to pursue a theme, epic or even story for which there is no longer a business case. It has to be said that while

Product backlog grooming and sprint reviews should guard against this happening by providing fast feedback from stakeholders, it is often the case that these activities focus on functional and user experience issues more than economic reviews.

More Thoughts on Granularity for ROI

Now that we have an approach for calculating ROI for Agile, it is worth considering the appropriate level of granularity further because our calculation approach involves cooperation, or at least coordination, between the development group and the business sponsoring the development. Various organizations implementing scrum have found it difficult to get commitments from the business for the day-to-day involvement necessary for the Product Owner role. This may be due to lack of resources or lack of understanding or some other reason embedded in the traditional corporate culture (and budget). Building and tracking business cases in the past was a business function or a PMO function. The PMO may no longer exist in an Agile organization so moving to ROI from business cases with Agile implementations will require some education and will need to recognize that the effort required will only be justified at quite high levels of aggregation – the assumptions required to operate at this high level of aggregation will seem quite contrary to Agile principles at first.

SAFe addresses this challenge by defining "Product" and "Portfolio" levels of aggregation above the "Team" level characteristic of Scrum. Business cases are made and long-running (longer than a typical project or release) "trains" of scrum teams are funded at the Portfolio Level.

One of our clients takes a different approach by using a hierarchy of story types in which high-level "Demand Stories" correlate closely with business cases at the business-development interface. These Demand Stories are then broken down successively into Customer Experience Stories, Component Stories, Engineering Stories and so on. Their experience with this approach has led to them building a rule into their story management system that only complete Demand Stories can be progressed through their Kanban stages. That is, all of the subsidiary stories must be complete before any one of them can be progressed because too often subsidiary stories were worked on in isolation but the expected business value could only be realized when all the subsidiary stories in a Demand Story were delivered. The phrase "orphan CE stories" barely hints at some of this issues that can occur if some such rule is not in place. But, again, many people think that such a constraint is in conflict with the principles of agile.

Conclusions

We believe that it is meaningful to calculate ROI for Agile "Projects" but only at an enterprise or portfolio level. Further, we have demonstrated that it is not a trivial exercise. Finally, we believe that the risk of funding long-running projects which turn out to have limited business value is much smaller with Agile than with Waterfall or other SDLCs.

.

Sources

- "Self-Funding Projects – A Benefit of Agile Software Development", Tara Hamilton-Whitaker, July 22, 2009, https://agile101.wordpress.com/tag/roi/
- "Foundations of the Scaled Agile Framework® Presentation, ©Leffingwell et al. © 2014 Scaled Agile, Inc. (with permission).
- "Steps for Calculating ROI on Agile Projects", Richard Banks, http://www.richard-banks.org/2007/07/steps-for-calculating-roi-on-agile.html
- "The Business Value of Agile Software Methods: Maximizing ROI With Just-in-time Processes and Documentation," 2009, by David F. Rico, Hasan H. Sayani, Saya Sone.