SolutionsIQ

The Business Value of Agile Transformation

By John Rudd

Overview

The potential benefits of full-scale Agile are enormous, although rarely fully realized. Many of the companies that adopt Agile are simply not prepared to make the far-reaching changes necessary to obtain the full benefit. Executive sponsor reluctance is understandable, since some of the most talked about Agile benefits, such as employee well-being and business adaptability, are also notoriously difficult to measure. However, there are many Agile benefits that can be and are being measured in traditional business terms. Our purpose here is to make the Agile sponsor's job easier by clearly demonstrating how specific Agile features link to tangible performance improvements. We discuss six business benefits that are available to any company that adopts Agile that can be measured using traditional financial and production metrics.

1. Reduced failed project risk
2. Reduced over-budget and late projects
3. Reduced waste
4. Improved return through early and frequent releases
5. Reduced write-off risk
6. Higher-quality software with fewer defects

There are many Agile benefits that can be and are being measured in traditional business terms.

1. Reduced Failed Project Risk

The traditional approach to project risk management is front-end loaded and control-oriented. Prior to project start, a charter determines what should be delivered, when it should be delivered, and how much it should cost. With the assumption that key risks and mitigations are now well understood, a detailed project plan is laid out to determine what is believed to be the most likely path to success. The mission of traditional project management is to incorporate controls that ensure that the project never deviates from this path over the course of the project's lifecycle. Value is received only when the project is completed. This approach to software project management has never been very successful. In fact for many years, it was the shame of the software industry that software projects on average performed much worse than projects in other industries. We now have a much better understanding of why this was the case.

First, software development is more of a design process than a construction process. One implication is that important characteristics of the solution cannot be known in advance and will emerge as the project unfolds. Second, over the last few decades the exponential rate of technology has created more and more volatile business conditions. Over the course of a year or two, conditions can change in surprising ways and render well-researched project plans obsolete overnight. Traditional risk controls were designed to keep project plans from changing. Yet we now see that, for projects to succeed, plans must continually adapt to changing conditions. Ironically enough, by resisting change, traditional risk management often leads to project failure rather than project success.

> 76% of those surveyed said they experienced reduced failed project risk after implementing Agile.

In the 9th Annual State of Agile Survey study conducted by VersionOne, 76% of those surveyed said they experienced reduced failed project risk after implementing Agile.[1] Agile mitigates failed project risk by:

- Using a project risk strategy that makes changes easy rather than hard.
- Building deliverable software in short iterations that allow for ongoing plan modification

- Reducing the cost of change by allowing and encouraging modifications to the specs (backlog) throughout the project lifecycle
- Establishing persistent teams organized and trained to employ design thinking, learn fast and respond rapidly
- Focusing on the minimal marketable feature set (MMF) to enable releasing to the market early

2. Reduced Over-Budget and Late Projects

Software development is a complex, organic process, which is inherently difficult to effectively estimate when all you have as a guideline are written specifications. Software specifications are notoriously inaccurate and incomplete and grow more undependable as they grow in size. The traditional approach is to complete an estimate for the entire release before the project begins. However, with no reliable way to accurately predict the cost or time needed, these estimates are largely unreliable. Traditional milestone segmentations of project schedules don't solve this problem, since there is no way to accurately measure the completeness of project artifacts until late in the project schedule. The result is poor project performance. Projects are either late with overrun budgets or contingency buffers are used to improve estimates at the cost of diluting project outcomes. Half of all systems projects overrun their budgets and schedules by 200% or more, and up to 40% of projects fail prior to completion, often due to the fact that, by the time the project is complete, the customer need has shifted to the point where the value of the project is diminished.[2]

Stakeholders can decide to stop the project early... without compromising the ability to release what has already been built.

Agile methods address these problems in a number of ways:

- By empirically measuring the time it takes to complete a short increment (often in the form of team velocity), you can project a fact-based estimate of the time and effort needed to complete the rest of the work.
- By completing all the work necessary to deliver a slice of requirements during the iteration, you can more accurately measure the percent of work completed. The fallibility of the milestone approach is avoided.
- Continuously re-estimating of work over

the course of the project encourages the use of the best information available and "truth telling" even if it contradicts initial estimates.

» Team artifacts such as "definition of done" are used to ensure that all necessary work is completed during each iteration.

» Frequent Sprint demos provide frequent customer feedback based on actual running systems, not a prototype or a design doc. This also demonstrates that real progress is being made.

» Shorter release cycles means shorter forecast time horizons which means more accurate estimates.

» Because estimation is more accurate, schedule and budget problems are forecast early, leaving more time to implement the proper mitigations.

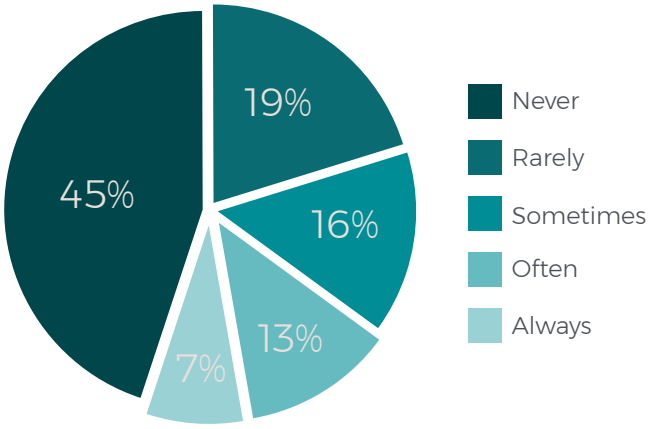
» Because the first market release is often much earlier in the process than under waterfall, the ability to modify or reduce project scope can result in under-budget projects.

A 2008 VersionOne survey indicated that more than 60% of respondents reported a project cost improvement through using Agile.[3] This is no doubt partially due to the fact that sponsors can decide to stop the project early—for example, if they do not forecast adequate ROI (see benefit 4 below)—without compromising the ability to release what has already been built.

3. Reduced Waste

Fundamental to Agile is the application to knowledge work of the very same Lean waste reduction principles that revolutionized manufacturing. Examples of Lean waste include work in progress, wait times, inventory, and rework, all of which have analogues in software development projects.[4] Furthermore the inherent uncertainty of software projects further increases the likelihood of waste. Just as in manufacturing, just-in-time software development implemented within the context of an incremental improvement program results in huge cost reductions over time, while compounding delivered customer value (see inset on next page).

One of the most effective ways to reduce waste is to avoid building the unnecessary. At The 2002 XP Conference, Jim Johnson from the Standish Group shared the results of their research: according to study respondents, 45% of software features were never used, and that 20% of the features provided 80% of the value.[5]



Rarely or never used features compared to all features released

Here are some examples of how waste is reduced in Agile:

» Reduced margin-of-error buffers, because estimates are more accurate
» Reduced wait-time between tasks and phases (e.g., testers waiting for developers to complete their tasks)
» WIP waste reduction
  · Increased efficiency and reduced cycle time through test automation and continuous integration
  · Use of just-in-time production principles such as small batch sizes (short iterations and short release cycles)
» Over-processing waste reduction
  · Less over-building and over-engineering, which tend to compensate for incomplete specifications
  · Frequent releases resulting in reduced customer demand to build features with dubious value
  · Elimination of long and complicated project plans with phonebook-sized, non-added-value design artifacts
» Incremental improvement
  · Incremental improvement built into team behavior through team retrospectives every week or two
  · Team empowered to make changes to improve quality and reduce waste

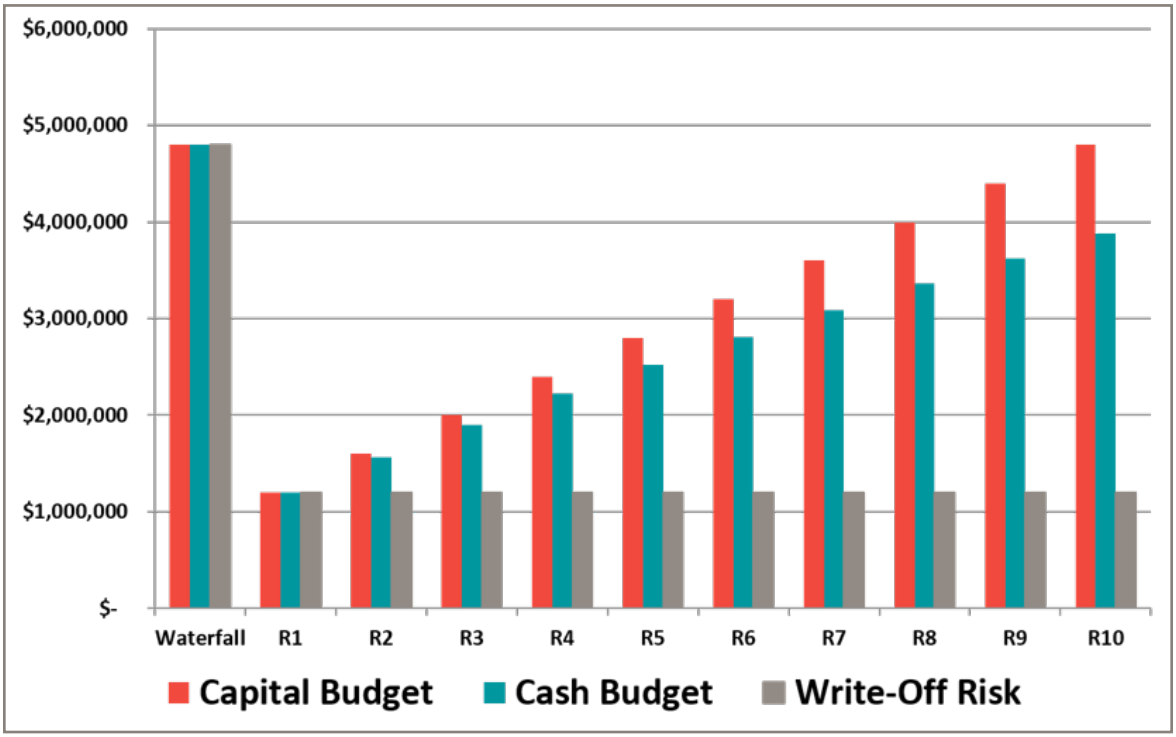
Just-in-time software development... within the context of an incremental improvement program results in huge cost reductions over time.

4. Improved Return Through Early and Frequent Releases

Increasing market volatility means capital investments must be reduced to correspond with shrinking and uncertain return horizons. In addition, the quicker you release, the smaller the capital investment you need to obtain a return on your investment. From a portfolio perspective, smaller capital investments means more investment opportunities and more portfolio risk diversification.

The chart below illustrates three ways incremental release increases financial performance when compared to the traditional "all or nothing" release approach of waterfall projects.

- Incremental capital funding conserves capital while ensuring earlier positive return.
- Cash requirements are partially offset by earlier earned revenue
- Write off risk is capped at the cost of an release increment (rather than the cost of the entire project)



Incremental release increases financial performance in three areas, compared to traditional Waterfall release

In addition, return can be further enhanced by the following factors, particularly when used in combination:

- Extending the project as long as there is market value to harvest
- Dynamically adding new features as new opportunities emerge over the course of the project
- Implementing the most valuable features first and ending the project when the forecast return on remaining features is no longer acceptable

The quicker you release, the smaller the capital investment you need to obtain a return on your investment.

5. Reduced Write-Off Risk

For a variety of reasons, many IT projects that are started are never released and the investment must be written off. Agile methods reduce write off risk through:

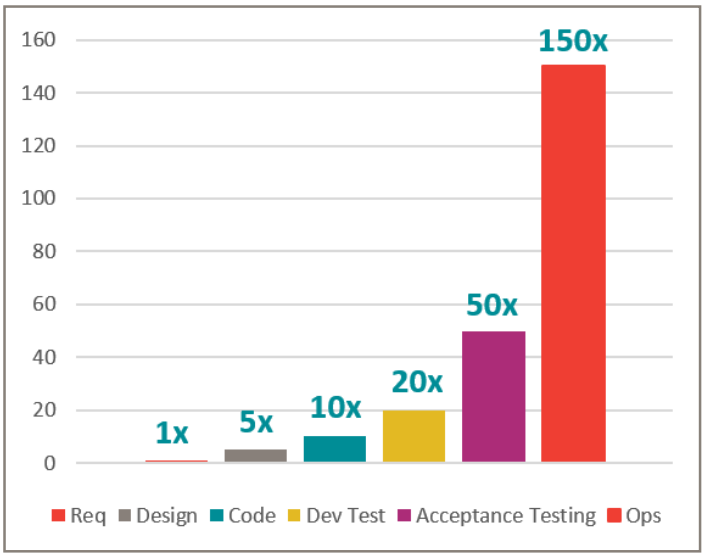
- Failing fast
  Iterative development makes it possible to determine sooner rather than later whether a project investment should be stopped.
- Early customer feedback
  Early release and frequent demos help surface information that helps determine how to reshape the project to save it or end it early if it can't be saved.
- Persistent teams
  Investing in persistent teams makes it feasible to quickly stop failing projects and transition to new projects with minimum startup costs.

The chart above outlining the financial benefits of incremental release also shows how Agile dramatically reduces write-off risk.

6. Higher-Quality Software with Fewer Defects

Barry Boehm demonstrated many years ago that the later in the production process you find a defect, the more costly it is to fix.[6] Complimentary Agile practices work together to provide a holistic software quality program. Taking steps to eliminate the cause of defects and fixing them as soon as possible is a core Agile principle that is embedded in every Agile practice. This reduces the cost of development and testing, as well as the total cost of ownership over the production life of the system. Further cost savings come from reduced re-work and higher customer satisfaction.

Examples include:



True cost of a bug fix

User story requirements definition

- User stories implement the highest standard for software quality assurance: testable requirements.
- Frequent customer demos
  Frequent demos and acceptance of incrementally completed work provides ongoing validation that the features implemented are implemented properly, which further increases customer satisfaction and reduces the need for re-work.
- Test-driven development (TDD)
  TDD, test automation, continuous integration and other extreme programming practices dramatically increase quality and the cost of testing by eliminating the latency between coding and testing to minutes or hours rather than weeks or months.
- Agile design principles
  Refactoring, emergent design and other Agile design practices ensure a robust code base that is easy to change without introducing new defects.
- Technical debt remediation
  Agile development practices slow the accumulation of new technical debt and can reduce the technical debt of legacy code bases.

» Iterative development
Shortening latency between requirements and systems integration testing to days from months

In addition, the Rico study showed quality improvements in over 60% of the companies surveyed.[4] Eighty-four percent of the participants in the 3rd Annual State of Agile survey said defects were down by 10% or more,[3] while this year, 78% of participants in the same survey said that enhanced software quality motivated their use of Agile approaches.[1]

Summary

Adoption of the Agile methodology has moved from what was largely grassroots initiatives to now mainstream approach to software development. This accelerated transition from traditional practices to the Agile approach was in many ways born out of necessity. The emergence of the speed of technological change and quick customer adoption has been disruptive to many markets, leading to organizational imperatives to (1) reduce time-to-market and (2) make smaller market investments. Often organizations reeling from this market disruption do not pause to focus on the tangible value that can be obtained by an Agile approach. Instead they simply push forward with an Agile initiative as a reaction to significant project failure or a loss of market share. While reduced cycle time in itself can sufficiently justify adoption of this methodology, it's imperative to pause for a second to consider the other attributes of Agile software development that can potentially be as valuable if not more. In this paper we have defined the six major benefits of Agile transformation in terms of tangible business value. To recap, they are:

1. Reduced failed project risk
2. Reduced over-budget and late projects
3. Reduced waste
4. Improved return through early and frequent releases
5. Reduced write-off risk
6. Higher-quality software with fewer defects

Careful consideration of all of these benefits and incorporating means of achieving them into your transformation roadmap will help ensure that your organization can reap the full value of an Agile change initiative.

[1] 9th Annual State of Agile Survey. VersionOne, Inc. 2015
[2] Breakthrough Technology Project Management. Lientz and Rea. 2011
[3] 3rd Annual State of Agile Survey. Version, Inc. 2008
[4] When You're Agile, You Get Lean. Rudd, Charlie. 2015
[5] http://www.martinfowler.com/articles/xp2002.html. Fowler, Martin. 2002
[6] Software Engineering Economics. Boehm, Barry. 1981

SolutionsIQ

The Business Value of Agile Transformation

Want to learn more?

Visit SolutionsIQ.com

Email: info@solutionsiq.com

Toll-Free: 1-800-235-4091

Direct: 1-425-451-2727