BPTrends COLUMN

The Agile Practitioner: Agile Risk Management

Tom Bellinson May 5, 2020

Over a decade ago now, I was a full-time business process consultant. Most of my work centered around preparing for enterprise level information technology projects. Back then, there were several studies that suggested that over 50% of these types of projects failed.

This was all before agile practices had taken hold. The reason many projects were considered a failure was because at that time, it was common practice to develop a detailed project plan (often in the form of a Gantt chart). This project plan specified not only the scope of work required, but also the duration of the effort.

As we have learned from those experiences, locking in both scope and timeframe is a recipe for failure. In fact, when researchers reviewed project success or failure, failure was defined as:

1. Did not achieve the expected project outcomes
2. The project took longer than expected
3. The project was abandoned due to major issues (budgetary, priority changes, etc.)

Agile practices are fundamentally designed to mitigate these types of project failures. If agilistas had their way, they would be provided with clearly defined "customer" outcomes and then business folks would step back and let the team(s) do their work.

Reality

Unfortunately for agile purists, the world doesn't work that way. Sometimes there are deadlines. Oftentimes these deadlines are externally imposed. Examples might include having something ready to launch at a major trade show, or having something ready for a regulatory due date.

Sometimes, those of us doing technical projects will be handed what appears to be a fixed scope. This comes in the form of a list of

expected outcomes (assuming the organization is supporting agile practices).

In this sense, those of us who work with or on agile technical teams may sometimes feel that nothing has really changed. If we continue to get handed work with fixed scope and fixed due dates, we are being forced to accept an agile anti-pattern.

Managing Risk

I work with several teams, but one in particular is subject to business-driven due dates. Furthermore, some of this work must conform to predefined industry guidelines that seem like a fixed scope to the uninitiated.

The reality is somewhat different in a truly agile organization. While the externally imposed deadline may be immovable and conformity to the guidelines is a required outcome, how the team delivers on those outcomes is very fluid. Teams often like to use the term "minimum viable product," a term coined or at least popularized by Eric Ries in The Lean Startup (a book I reviewed a while back here on BPTrends), to describe that first useful product delivered to customers. Teams, in conjunction with their product manager (or product owner in agile parlance), should evaluate different ways to deliver outcomes that can meet a fixed deadline. Whenever possible, the least understood work should be prioritized to be explored as soon as possible in order to expose high risk elements of the project early.

With these two concepts in mind:

1. Explore different value delivery options
2. Expose high risk elements early

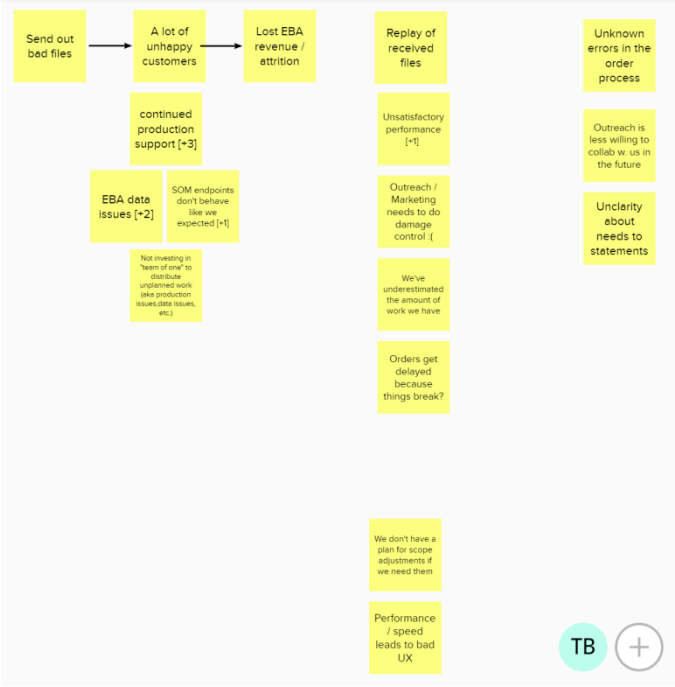
Let me share an example of this type of process from one of my teams. As of this writing, our team finds ourselves, like so many others, working from home during the Covid-19 pandemic. Normally, we might use a whiteboard to brainstorm, but we find ourselves using tools like Jamboard or Mural. In this case, the team has chosen Mural, which allows us to use a variety of templates to quickly develop a shared understanding of our situation.

At ITHAKA, we have the good fortune to have fully cross-functional teams. Thus, our product manager, quality assurance (QA) lead, developers, and user experience (UX) professional are all full-time

members of the team and fully engaged in our process. Having all of these different perspectives is incredibly valuable.

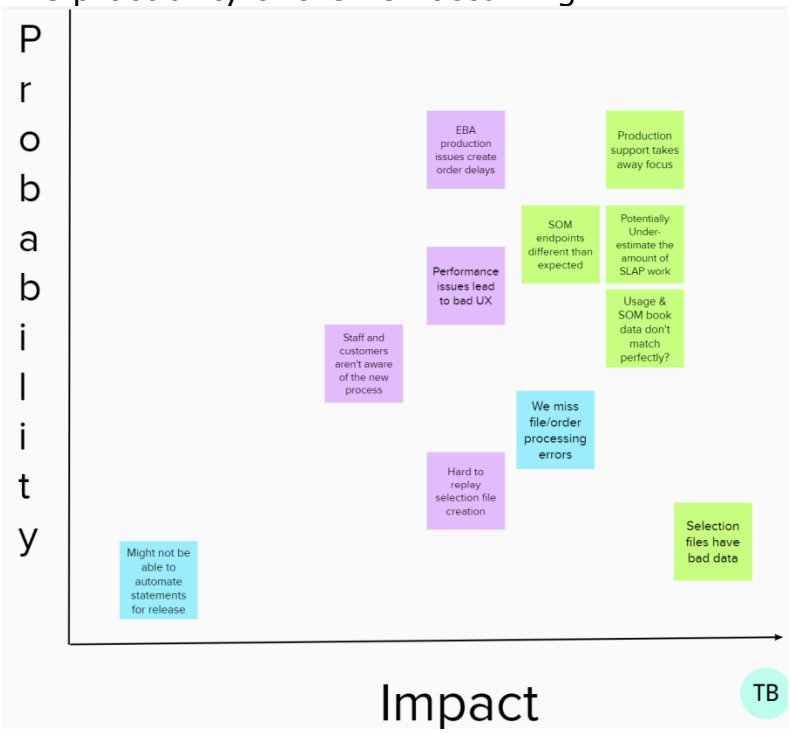
Our UX person and product manager bring a deep understanding of what our customers (or users as we generally refer to them) need and expect. They also understand where the improvement opportunities lie. Our developers bring the expertise to figure out how to unlock those opportunities in different ways. Finally, our QA lead brings his unique perspective on risk.

Step one in our process was to have an open brainstorm of potential risks. We all wrote cards for each risk we could think of. That looked something like the image to the right. The cards were not so neatly organized as we wrote them, but after our initial round of brainstorming, duplicates were combined and some wordsmithing for clarity happened.
Next, we classified these cards by risk level. As we discussed risk, we realized that there are essentially two components to risk:



Step one in our process was to have an open brainstorm of potential risks. We all wrote cards for each risk we could think of. That looked something like the image to the right. The cards were not so neatly organized as we wrote them, but after our initial round of brainstorming, duplicates were combined and some wordsmithing for clarity happened.
Next, we classified these cards by risk level. As we discussed risk, we realized that there are essentially two components to risk:

1. The impact of the risk
2. The probability of the risk occurring



We had initially just put each card in a high, medium or low category, but this proved to be inadequate. Realizing this, we switched to a chart representing the two factors.

This process quickly exposed a few high probability / high impact elements of the work that we could explore now. In this case, the things that we uncovered also happen to be critical to delivering the value of this initiative. This means that we did not have the option of deferring this work until after the initial launch.

Making some work a "fast follower" is often a good option to deliver value on time by deferring riskier efforts until after the deadline. Sometimes, without deferring functionality, we can scale back the value while still making advancements. For example, our plan is to receive an input file from the user and automatically process it without intervention. One option that we have discussed if we cannot get that far before the deadline is to gather the input files from the user automatically, but inject some manual intervention by our team to process those files until we can fully develop a reliable approach to automatic processing.

The user will likely experience a longer delay in receiving the feedback from the processed input file, but the new process will still be easier and faster than before the new functionality was put in place. Most of the burden of this compromise is on the team's manual intervention, which makes us very motivated to work out the automation quickly after launch.

We also uncovered some risks associated with new areas of work for which we have little or no experience. An example of this would be the input file. We have extensive experience generating Microsoft Excel files to be delivered as report outputs. However, we have never had to receive an Excel file into which a user has entered data for us to incorporate back into our system. During our interviews with users about how they might use the Excel input template we provide to them, we learned that it is quite likely that the file that is returned to us may not be consistently formatted in the way we would like. The file could have been split up and passed around to multiple constituents and then put back together such that the resemblance to the original is not precise. We do not yet know the range of what we may get back or what tools we have to interpret these variances accurately.

Recognizing this risk, we are moving quickly to explore our options and evaluate approaches to protecting the format while still allowing users the flexibility to effectively execute their own processes. Through our learning, we will likely find multiple strategies. We will need to pick one to try at launch, but having spent the time to go through the learning process, we will have other options waiting in the wings to try as necessary once we get experiences with real users in production.

To Summarize

One of the fundamental benefits of the iterative nature of the agile approach is to reduce the blast radius of risk. There are always risks, but if we can iterate quickly with real customers in the loop, we can learn the lessons that can only be gained in the field.

When a deadline looms large, product teams must have the freedom to deliver on the required outcomes for the customer in a highly flexible manner. Organizations that dictate how features must be implemented from above, will stifle a team's ability to creatively unlock user value. Of course, this means that leaders need to understand the difference between defining outcomes versus features. Outcomes are defined in business terms whereas features are technical in nature. It

is unlikely that anyone outside of the team actually implementing features will have a better understanding of how to deliver them.

Correspondingly, product teams must develop the ability to understand and process the implications of outcomes and user value, such that they can creatively translate these into optional delivery models in the form of various feature sets. When teams have this capability and are empowered by an organization that embraces an agile culture, risks can be easily mitigated with a little planning, preparation, and creativity.