Thinking Fast and Slow for Project Management

Murray Cantor, John Heintz, Steven Sherman
Introduction

Novelty, Uncertainty, and Risk

In this paper, by project risk we mean the probability of missing the target date and/or budget. Projects in any domain range in innovation. They can range from simple routine maintenance to adding a feature to an existing system or building to delivering something with an entirely new design. For example, there was a lot of innovation in building out the Sydney Opera House. There is not much innovation in building yet another small strip center.

![The Risk Curve](image)

*Figure 1. The risk curve*

A key insight is
- The more innovative the effort, the less is known for certain
- Without complete information, the team or anyone else is able to make firm predictions as to the cost and schedule of the project
- The inability to make firm predictions mean there is project risk.
This general principle works across domains: Software, Construction, System Integration, etc. The relationship between risk and novelty is captured in the figure 1:

- Routine maintenance efforts have complete information about what is needed to be done by sufficiently skilled workers.
- When adding a new feature, there may be uncertainty about what is really required by the stakeholders and how the system or product needs to be modified.
- When building a new platform or newly designed skyscraper, or new weapon system, or anything else, there is uncertainty about the stakeholder requirements, the architecture, and level of effort needed to complete the effort.

As we will discuss below, the uncertainty (risk) is reduced by gaining information through by iterative efforts, and focusing activities not only be completing task, but also by actively seeking to fill in the missing information, and resetting the activities to react to what has been learned.

Uncertainty, Risk, and Reward

The concept of risk/reward is well known by investors. Simply put, to get good returns, one needs to make riskier, less certain investments. However, one should not take the risk without knowing the odds of success. Even bet, if you can take action to improve the odds, all the better. This is also true for development. Generally, the ability to assume risk and deliver in the face of uncertainty enables the organization to deliver more value.

Reasoning about Risk

Chartering and managing an innovative, risk project entails reasoning well about the probability of success. Experience has shown that almost everyone has limited ability to determine that probability from the information at hand. This takes applying analytic techniques from applied probability, i.e. ‘slow thinking’. This is turn requires appropriate software tooling. We will discuss this in more detail below.

Agile and Uncertainty

Agile was developed to successfully deliver projects in the face of uncertainty. After decades of struggling to treat software projects like factories or construction projects, along with more positive outcomes from iterative approaches such as spiral development, extreme programming (XP), and the unified process, in 2001, thought leaders of the development community met and published the now-famous Agile Manifesto, a set of principles that, when followed, support team learning, continuing stakeholder interaction, and experimentation to
steer the project to successful completion. Consequently, Agile has had a measureable positive impact on project success.

However, Agile has also battled significant challenges with management. Managers crave and need predictability, but Agile practitioners are adamant that firm predictions are impossible. Recognizing the need and difficulty of resolving this dilemma, we have formed Aptage, a company that delivers services to facilitate the right conversations and decisions among the Agile team, its manager, and its stakeholders. Our services recognize that the solution entails dealing with probability. This, in turn, requires a combination of fast and slow thinking.

**Modern Insights**

There have been several key ideas that underlie good management of uncertain efforts. Let’s examine each of these in more detail.

“Plan Your Work; Work Your Plan” is Misguided

Once upon a time in the story of project management, we were told to “plan your work; work your plan.” The concept was to gather all the information needed to plan the work in fine detail. If you built the right plan, the project should go like clockwork. Ideally, there were no decisions to be made. If things didn’t go as planned, you were a bad project manager. Following this edict is known as following the “waterfall” process.

Also, at the time, software was considered an immature field because almost no software projects went as planned. The problem was not with software as a discipline. The problem was that the project management practices at the time did not consider the amount of invention and innovation required to deliver the product. Indeed, innovative engineering projects to date have rarely kept to plan. Some well-known civil engineering examples include Boston’s Big Dig and the Denver International Airport. A big IT example is initial failure of the Affordable Care Act (ACA) portal. There are many others.

Goal-Driven PDCA is The Better Alternative

Over the last decades, there have been several alternatives to the waterfall process. The essence of all these processes can be found in the PDCA (Plan-Do-Check-Act) cycle, shown in Figure 2.
The idea behind the cycle is that once you have created the initial plan, you would:

1. Start carrying out the plan
2. Check to see if you are progressing toward your goal
3. Based on what you learn in the check step, take some action to better meet the goal
4. Update the plan

You periodically repeat this cycle, steering the project to successfully meeting this goal. Each cycle may be called an iteration, or in Agile, a sprint. This method works for innovative efforts when, during each cycle, you fill in the needed information about the effort. What information you gain depends on the project. Typically, we monitor and measure:

- Our understanding of the stakeholder requirements of the deliverable
- The actual performance of the team (the velocity)
- Our understanding of the scope of the effort

Actions might include scope management, rescheduling the completion date, or adding resources. (Note that Agile development is a set of methods for the PDCA cycle for software.) The key insight to PDCA is that in each cycle, you should learn something that decreases uncertainty and risk. Good project managers do this instinctively, initially working on the “things that keep them up at night” rather than the easy, low-hanging fruit. While PDCA makes good sense, we are left with the question, “How does one instrument the method?” The Check and Act steps in the cycle are critical to success. In an effort that requires learning, it comes down to what to check and how to act. For instance, for an effort with some risk, we need to check the likelihood of meeting goals (say, on-time delivery).
No machine can drive a PDCA project. The role of instrumentation is to help people draw the right conclusions from the information and take the best-informed action. To better understand how a tool might help in this process, let’s step back and consider how people reason.

People Use Fast and Slow Thinking
Lately, there is a resurgence of interest in Daniel Kahneman’s highly acclaimed book *Thinking Fast and Slow*. Kahneman and his colleague Amos Twersky have studied how we make decisions — whether we’re talking about the US presidential election or estimating the date that our product will be out the door. The insight of Kahneman and Twersky is that people do both fast and slow thinking using different, separate cognitive systems. The insight of Kahneman and Twersky is that people do both fast and slow thinking using different, separate cognitive systems.

Fast thinking is subconscious and spontaneous. Examples of fast thinking:
- Seeing that an object is at a greater distance than another
- Localizing the source of a specific sound
- Displaying disgust when seeing a gruesome image
- Having a “hunch” the poker opponent is bluffing
- Just “knowing” that global warming is a hoax
- Believing the polls were wrong because Trump won

Slow thinking is deliberate, logical, calculating. Examples of slow thinking:
- Solving an algebra word problem
- Computing the odds of having the winning poker hand before deciding whether to fold, call, or raise
- Evaluating the climate change evidence and deciding whether it is likely enough to act
- Realizing that Nate Silver’s 28% forecast of Trump’s victory puts it well within the bounds of possibility

For a good, quick explainer about fast and slow thinking, watch this video.

You might imagine that slow thinking is superior. However, both are important. For example, imagine you are strolling through the savannah and hear some loud rustling behind you. Are you going to compute the odds that it is a lion or just get the hell out of there? Relying on fast thinking at the exclusion of slow thinking results in overconfidence and lots of false starts. Slow thinking, on the other hand, can verge into “analysis paralysis” and being eaten by lions.

Just like excellent poker playing, deciding how and whether to act in an innovative effort requires both systems of thinking. You need slow thinking to update your current beliefs with recent learning and then use fast thinking to act based on your experience and intuition.
pointed out in Thinking Fast and Slow, slow thinking requires applied probability, even Bayesian reasoning. People, on their own, are generally not good at this, which is why quantitative analysts exist and use lots of tools.

**Modern Trends in Dealing with Uncertainty**
A typical conversation when planning a project can go like this:

**Manager:** "How long will it take to complete this work item?"
**Worker:** "Beats me. I have never done this before."
**Manager:** "I need to be able to tell our stakeholders when to expect the deliverable."
**Worker:** "Too bad, I really shouldn’t guess."

And a bad time is had by all.

So we have a dilemma: The manager is asking for a forecast she needs that the worker, for good reason, is loath to make. In fact, the worker, using his fast thinking, is emotionally reacting to the request.

So how would someone using slow thinking approach the dilemma?

First, realize that the time-to-complete is uncertain and thus the best we can do with the information we have is to determine the probability distribution of, say, the time to complete. Following the insight of author Douglas Hubbard, the worker in our scenario may not be able to answer the manager’s question, but he is not completely ignorant. Suppose the conversation instead took this turn:

**Worker:** "Beats me: I have never done this before."
**Manager:** "Fair enough. Let’s get some bounds. Can you get it done in a year?"
**Worker:** "Of course."
**Manager:** "More seriously, what is the longest you think it would take?"
**Worker:** "Given the amount of distraction and if I had to code from scratch, maybe 7 months."
**Manager:** "OK. Suppose you are left alone and you can find some code to reuse, how long?"
**Worker:** "Well, in that case, it could be done in 2 months."
**Manager:** "Thanks, and what is your gut feel?"
**Worker:** "I am not sure – say 4 months."
**Manager:** "Great. What can we do to reduce your uncertainty?"

A lot has happened in this conversation. The manager respected the conversation by not insisting on an unreasonable estimate, and now has some idea of the level of effort and the worker’s uncertainty.
The slow thinking approach is to capture the wisdom of the team as a probability distribution. In this case, a simple triangular distribution works well. In a triangular distribution you use low, expected, and high values, where:

- The probability of the value being below the low is 0
- The probability of the value being above the high is 0
- The value with the highest probability is the expected

Figure 2 shows examples of how triangular distributions are represented graphically.

![Three triangular distributions for backlog item story points (the expected and high or low values can be the same).](image)

Another version of this perspective is found in *Superforecasting*, a book by Philip Tetlock and Dan Gardner. The idea is that if you were to ask several people for a value about which they, collectively, are uncertain, you will get a range. So the question to ask is, “What should you do with the answers?” The usual fast-thinking response is to take the average, the so-called expected value. As pointed out by Tetlock and Gardner, however, you are throwing away crucial information — the team uncertainty:

> “Some reverently call it the miracle of aggregation, but it is easy to demystify. The key is recognizing that useful information is often dispersed widely, with one person possessing a scrap, another holding a more important piece, a third having a few bits, and so on.”

An instance of these ideas in action could be found in a variation of agile planning poker. In Agile:

- Iterations are called sprints
- Small work items, typically completable in a single sprint, are called stories
- Larger work items that take more than one sprint to complete are called epics and/or features
- Agile teams decompose their epics into stories which are assigned to sprints
- The level of effort assigned to stories and epics are called story points
- The number of story points completed per sprint is called the team velocity
At each sprint, you track how many story points have been completed (the burn down) and how many are remaining and plan accordingly.

Unlike manufacturing with its consistent processes, in software development, every story is different, and there is no one-story-point-size-fits-all. This is where planning poker comes in. In a team meeting, each member makes a “bid” on what his or her (thinking fast) number of story points is to the story. There is one wrinkle: To avoid false precision in the bids, many teams choose bids that are restricted to the set \{1,2,3,5,8,13,20,40,100\}. This is a common but far from necessary process. There is no fundamental reason why a member could not choose 10 points for a story, or 60 stories for an epic.

In a planning poker session, the team meets, goes through the backlog, selects a work item, and, after some discussion, chooses a consensus story point value for the story. Sometimes the team uses actual cards to bid the values. For example, for a given story, some might bid 3, some might bid 5, and others 8. The team, after some discussion, chooses 5. Some members of the team may have doubts, but they all agree to move on. This ignores the Superforecasting insight. There is some possibility the outliers are right. The one consensus value entails throwing away important information.

Let’s approach this process with some slow-thinking discipline. The assignment of story points to stories is not an exact science but an expression of beliefs. Following Hubbard and, in fact, the entire Bayesian school of reasoning, these assignments should be expressed probabilistically, again using triangular distributions. The actual project plan then requires combining these distributions in some meaningful way. It turns out that people’s intuitions on how these combine are not so good. Fortunately, we have slow-thinking methods that apply.

**An Example: Agile Management**

Agile management provides a good setting for fleshing out how thinking fast and thinking slow can work together. Here is a succinct overview of agile project management:

- A team creates a backlog of stories and features not yet decomposed into stories.
- These stories are sized with story points.
- The team sets the sprint length, usually one or two weeks.
- The team also settles on its planned velocity, the number of story points that can be completed in a sprint.

The team velocity depends on many variables: team size, skill set, available time, dependencies, and so on. Experience has shown that velocity does not settle into a stable value but has a large variation. So it too is best described by a probability distribution. Given a set of stories, the equation for the number of sprints it will take to deliver the stories is:
The equation of the time to complete (duration) the set of stories is:

\[ \text{Duration} = \frac{\text{Total Story Points}}{\text{Velocity}} \times \text{sprint length} \]

The planner adjusts the plan by either setting the number of sprints or, more commonly, choosing the stories from the backlog by priority, to reach the number of stories that can be completed in the time allotted.

Seems straightforward enough. However, for most real-world projects, the total story points and the velocity are uncertain. Hence, the number of sprints and duration are uncertain. And so, “plan your work; work your plan” cannot work.

**Finding the Likelihood of Success**

The fundamental question is whether your team is likely to deliver the stories your stakeholders need in the expected time, given the uncertainty. This is where slow thinking is necessary. We need to express the uncertainties as probability distributions and combine the distributions to get a sound measure of the likelihood of on-time completion. Figure 3 shows an example of distribution of the duration.

In Figure 3, the green area of the distribution is the probability of being on-time or early. The red is the likelihood of being late.

Computing the distributions in Figure 3 entails:

- Capturing the backlog items sizings using the distributions shown in Figure 2.
• Determining the probability distribution of the team velocity given the team’s beliefs and history using a Bayesian learning algorithm.
• Combining the sizings and the velocity distributions to an overall distribution using the above equations (using Monte Carlo simulation to do the arithmetic of the distributions).

Implementing this process involves using well-established, compute-intensive applied probability algorithms. We at Aptage have developed easy-to-use tools that implement the above process.

Using the Distributions
In a well-managed Agile effort, the red area (of Figure 3) will decrease to zero over the project’s duration. By the last iteration, the red area should be close to zero. When dealing with interesting projects with uncertainty, teams should augment the Agile practices listed above with:
• Reaching an agreement between the team and stakeholders on the amount of initial uncertainty to accept
• Focusing the sprints on not only burning down the backlog, but also burning down the risk (the red area)

A way to visualize the team’s progress in working off the risk is a risk burndown chart (see Figure 4).

Figure 4 The same project after 2 iterations, entering iteration 3. (Note the uncertainty in this figure has been reduced from the 41% in Figure 3 to 17%.)

In the risk burndown chart, the x-axis is the iterations (sprints) of the project and the y-axis is the likelihood of being late (the red area). The green area in the burndown chart shows the acceptable risk at each sprint. The initial risk is where the green area meets the y-axis. Note the green area slopes down to zero at the planned release sprint (there should be no uncertainty at the target date). The line shows the current risk computed at that sprint. If the line goes above the green, the project risk is above the agreed-upon acceptable risk, and the area above shows red.
Thinking Fast and Slow for Project Management

Figure 5 Three project risk histories

Figure 5 shows three examples of project risk burndown histories. The first example shows a project with little initial risk that was quickly worked off. In this case, the project might be overstaffed. In the second project, the team discovered after one iteration that the project was even riskier than initially thought and they never took the necessary actions to fully recover. The third project had a surprise in the fifth sprint (some feature took longer than expected), but the team took the right actions to get back on track.

Now we see the power of thinking slow. Using triangular distribution and burndown charts, the team and stakeholders have readily available views of the risks of the project on an ongoing basis. This gives them the information they need to continually be truly Agile by applying the PDCA cycle at every sprint (see Figure 6).

Making it Real

Aptage, Inc. • 7301 Ranch Road 620 N, Suite 155-275, Austin, TX 78726 • www.aptage.com
In summary, project management in the face of uncertainty requires slow thinking, and slow thinking requires tools. However, to be useful and provide the hoped-for improvements and support next-gen Agile management, not only must these tools deal with the inherent uncertainty of interesting projects, the tools must also be easily adoptable by:

- Using data teams already have
- Not adding any extra effort for the team
- Enhancing, not interfering with Agile practices (e.g., planning poker)
- Being easily configurable to the team’s lifecycle and terminology

A further requirement is that the tools provide visualizations that don’t require a math degree to understand. To that end, Aptage has worked closely with a variety of beta customers and Agile consultants to provide the first offering that meets these requirements and helps teams have better confidence and predictability in the face of uncertainty. Aptage has a cloud-based set of services that meets these criteria, and beta and paying customers have confirmed the hoped-for benefits. While these are early days and we are still adding capabilities, we are pleased to report that the tools do indeed help the teams, management, and stakeholders communicate effectively and make good decisions with less contention and drama.