Thinking Fast and Slow for Development

Next Generation Agile Management

Murray Cantor, John Heintz, Steven Sherman

Aptage
7301 Ranch Road 620 N, Suite
155-275, Austin, TX 78726
www.aptage.com

Interesting Projects are Uncertain

Software can vary from very creative to very routine. Routine work is dealing with small code updates that is especially common with the modern continuous delivery environments. Creative development can involve adding entirely new features to an existing system or even a new system. These are less common but can, when successful, add more value.

Modern Insights

There are several key ideas that need to underlie good management of uncertain efforts:

'Plan your work and work your plan' is misguided

When I started project management, we were told to "Plan your work, work your plan." The concept was to gather all the information you 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, I was 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. However, the problem was not that it was software. 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. In fact, any innovative engineering project rarely kept to a plan. Some well-known civil engineering examples include the Boston Big Dig and the Denver Airport. A big IT example is initial failure of the ACA portal. There are lots of 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 Plan Do Check Act (PDCA) cycle.

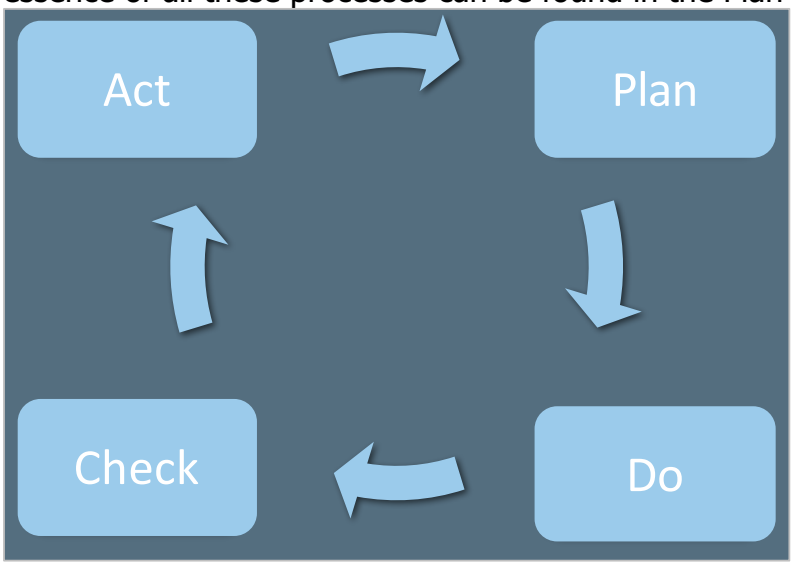

Figure 1. The PDCA method

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 to 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 ascertain the needed information about the effort. What information you gain depends on the particular project. Here, 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

After P, D, and C, Actions to take might include scope management, rescheduling the completion date, or adding resources. Note that Agile development is a set of methods for the PDCA for software.

The key insight to PDCA is that in each cycle, you should learn something that decreases the 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 Aptage's role, 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, *Thinking Fast and Slow*. For example, see Michael Lewis' *The Undoing Project*. Kahneman and his colleague, Twersky, have studied how we make decisions - whether we're talking about the 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.

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 the 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

(Here is a good quick (4min) explainer video about fast and slow thinking)

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 over-confidence 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. As 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 quants exist and use lots of tools.

Dealing with Uncertainty

A typical conversation when planning a project can go like a 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."*

And a bad time is had by all.

So, we have a dilemma: The manager is asking for a forecast that the worker, for good reason, is loath to make.

So how would someone using Slow Thinking approach the dilemma?

First, realize the time-to-complete is uncertain and, so, 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 Douglas Hubbard (*How to Measure Anything*), the worker in our scenario may not be able to answer the manager's question, but 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 of has happened in this conversation. The manager respected the conversation by not insisting on an unreasonable estimate, and 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 *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

Graphically, triangular distributions are represented like:

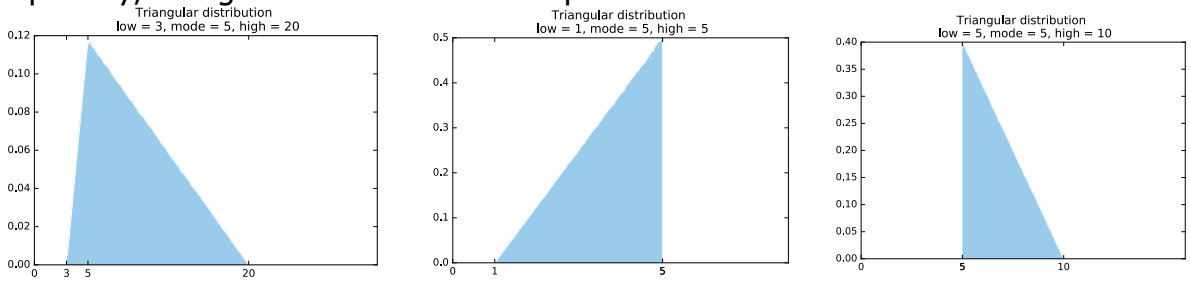

Figure 2. Three triangular distributions. In some cases, the expected and high or low values can be the same.

Another version of this perspective is found in *Superforecasting, The Art and Science of Prediction* 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 answers*? The usual, Fast Thinking, response is to take the average, the so-called 'expected value.' As pointed out in *SuperForeasting*, 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."***
>
> ***Tetlock, Philip E.; Gardner, Dan (2015-09-29). Superforecasting: The Art and Science of Prediction (p. 73). Crown/Archetype***

An instance on 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

Unlike manufacturing with its consistent processes, in development, every story is different, and so 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 their (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 a set {1,2,3,5,8,13,20,40,100}. This is a common but far from necessary process. There is no fundamental reason a member could choose 10 points for a story or 60 stories for an epic.

In a planning poker session, the team meets, goes through the backlog and selects, 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's, some 5's and maybe a couple 8's. The team, after some discussion choose 5. Some members of the team may have doubts, but they all agree to move on.

This ignores the *Superforcasting* insight. There is some possibility the outliers are right. The one consensus value entails throwing away import information.

Let's approach this process with some thinking slow 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, people's intuitions on how these combine are not so good. Fortunately, we have thinking slow methods that apply.

Modern Applications in Agile Management: Instrumenting PDCA

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 program management:

Notionally, managing an agile project of creating:

- 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 a large number of variables: The team size, skill set, available time, dependencies, etc. Experience has shown that velocity does not settle into

a stable value but has a large variation. So, it too can only be described by a probability distribution. To expect the velocity to stabilize is simply unreasonable.

Given a set of stories, the equation for the number of sprints it will take to deliver the stories is:

$$Number\ of\ Sprints = \frac{Total\ Story\ Points}{Velocity}$$

The equation of the time to complete (duration) the set of stories is:

$$Duration = \frac{Total\ Story\ Points}{Velocity}\ x\ sprint\ length$$

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.

This seems straightforward enough. However, for most real-world projects the total story points and the velocity are uncertain. Hence:

- The number of sprints is uncertain
- 'Plan your work/work your plan' is inappropriate

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 combine all the distributions we have, to get a sound measure of the likelihood of on-time completion. Figure 3 shows an example of distribution of the duration.

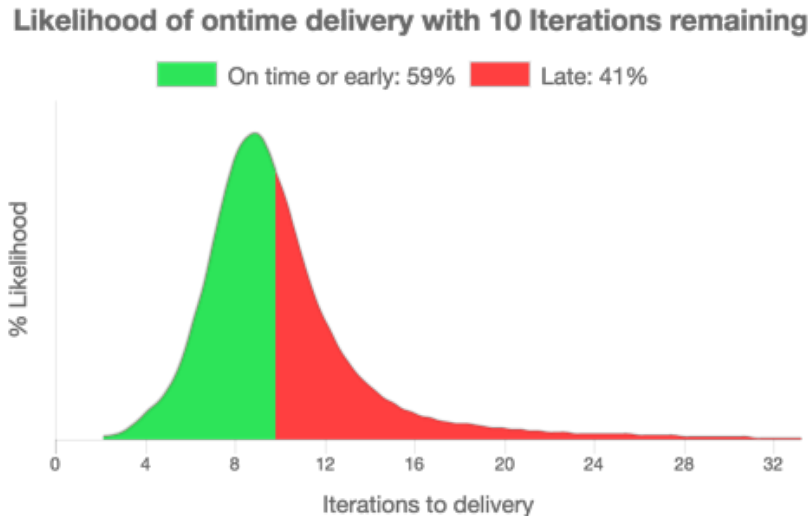

Figure 3 The initial time to complete for a plan with large uncertainty.

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.

In a well-managed agile project, the area of the red will decrease to zero. The PDCA cycle is instrumented by measuring the likelihood of being late.

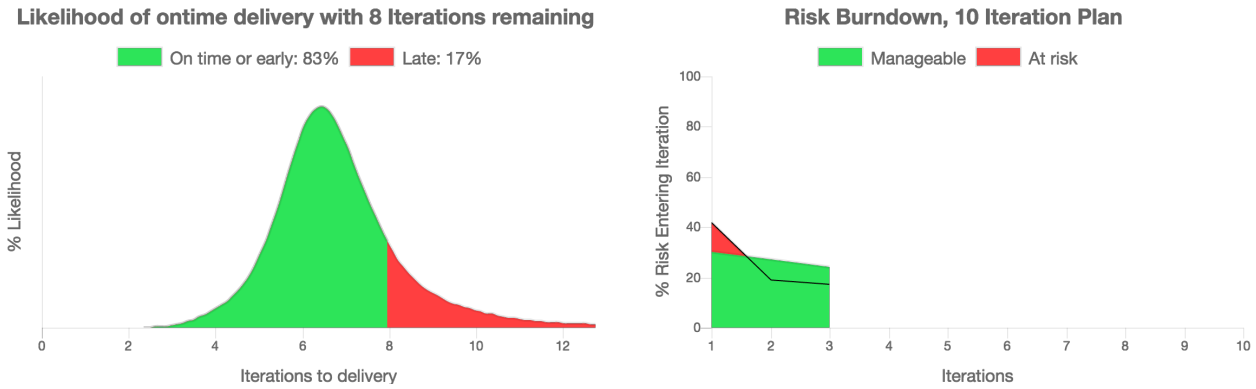

Figure 4 The Same project after 2 iterations, entering iteration 3. Note the uncertainty has been reduced from 40% to 17%.

Computing the distributions of Figures 3 and 4 entails:

- Determining the probability distribution of the team velocity given the team's beliefs and history
- Combining the sizings and the velocities distributions to an overall distribution.

Now we see power of thinking slow. No one, without tools, can assess the uncertainty of delivery or how that uncertainty evolves over the lifecycle of the program. Aptage provides those tools

Integrating Fast & Slow thinking So You Can Take the Right Action

There are two key principles for successful development project management:

1. For years, development experts have promoted rightly some variant of the PDCA cycle.
2. We have also been told that management processes should be instrumented. The Lord Kelvin maxim: Lord Kelvin is often quoted in this context:

"I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of science, whatever the matter may be".[1]

[1] Lecture on "Electrical Units of Measurement" (3 May 1883), published in *Popular Lectures* Vol. I, p. 73

We, at Aptage, believe both. Our mission is to take advantage of probability-based modern machine learning to provide state of-the-art instrumentation for PDCA style development (e.g. Agile).

Aptage is for those responsible for delivering projects with uncertainty. For those who experience the negative economics of missed deadlines and customer disappointment, conflict and drama with development teams, and have a need to improve development organization efficiency.

Aptage provides cloud based Artificial Intelligence enhanced decision support tools that integrate with customer's existing data to provide sophisticated, yet simple to use visualizations for managing development in the face of uncertainty.

Unlike larger tool sets that merely report on progress of plan, Aptage helps visualize and objectively quantify the risks inherent in the plan, and provides likelihood of outcomes and simulations of viable paths forward at every iteration of development.