

The #1 Blind Spot That Puts Agile Teams At Risk

How to Save Your Sprint in 3 Simple Steps



 **LiquidPlanner**[™]

The #1 Blind Spot That Puts Agile Teams at Risk

How to Save Your Sprint in Three Simple Steps

Overview

Delivering quality product on time and on budget is a tall order. There is no doubt it requires a talented team. It also requires a consistently clear view of the work effort and progress throughout the sprint. Unfortunately, too many teams rely on estimation methods that obscure rather than reflect reality. And bad estimates create a hazardous blind spot that can easily compromise sprint success.

This paper looks at some common pitfalls of sprint planning, namely bad estimation and inadequate scheduling tools. We discuss how these pitfalls create dishonest schedules and sabotage sprint success. We introduce the online project management application, LiquidPlanner, as a solution for Agile development teams seeking to prevent deadly blind spots with good estimates and honest schedules that optimize sprint planning and execution.

First, let's clarify what we mean by honest vs. dishonest schedules.

Your schedule may be lying to you if:

1. It is fed with **bad estimates**
2. It is **unable to adapt to change**

So, what makes an estimate bad?

Bad estimates all have one thing in common: they don't accurately reflect reality. And if you feed a sprint plan with unreliable estimates, you get an unreliable schedule. As they say: garbage in, garbage out.

Three kinds of bad estimates are particularly bothersome because they are so widely used by development teams of all shapes and sizes.

Bad Estimate #1 – Estimates based on story points

That story points are so widely used for estimating time in Agile development is a bit of a mystery. Why? **Because story points don't translate to schedule time.**

In fact, they don't have anything to do with the way real people even think about time. This means that teams have to establish their own translation guidelines for story points, and this thwarts efficiency and invites misinterpretation. It also makes it difficult to collaborate with stakeholders who don't use or understand story points.

Bad Estimate #2 – Blatantly unrealistic estimates

Development teams are under a lot of pressure from business owners who want to ship product as quickly as humanly (or not humanly) possible. To get business owners off their backs, developers tend to over-commit and enter overly optimistic (read: unrealistic) estimates.

Unrealistic estimates create problems for everyone.

For starters, they create idealistic schedules that look good on paper but have little utility or relevance in real life. And, since an overly optimistic estimate provides a best-case-only scenario, there is nowhere to go when a delay creeps in, which means the whole schedule gets botched. Finally, the expectation of business owners that optimistic is better than realistic can stress the capacity of development teams to an unhealthy breaking point, which threatens both morale and product quality.

Bad Estimate #3 - Estimates with hidden buffers

Adding a buffer to an estimate provides breathing room in the schedule. This accounts for the uncertainty of the estimator as to exactly when the task will get done given that, in development, the exact effort required for a given task is often unknown. Developers tend to like buffers; management does not.

The real problem is that a buffer is almost always hidden; no one really knows how big it is or what impact it has on the schedule. It is merely *implied*. Because of this gray zone, it is next to impossible to respect the buffer. And when the buffer is not acknowledged, much less respected, the schedule becomes compromised and iteration targets are easily missed.

The bottom line is this: a bad estimate may be overly padded, arbitrarily defined, or unrealistically optimistic, but it is always a danger to your schedule.

There's one more important point about bad estimates.

And that is that they are conveyed in single points of time. This, too, jeopardizes sprint health. As we saw with the best-case-only scenario above, single-point estimates (such as 4 days, 9 hours, or 13 story points) don't have a built-in margin of uncertainty to account for the unknown. They are rigid, fixed points veiled in a cloud of obscurity, and they create rigid, unreliable schedules.

This is a big deal, because rigid schedules are inherently anti-Agile.

One reason Agile favors “individuals and interactions” over “processes and tools” is that traditional project management tools are too inflexible for its dynamic, collaborative nature. Unfortunately, too many Agile teams are unwittingly using tools that work against them by locking them into bad estimates and rigid schedules. Or, if they reject tools altogether, they are left without much of a schedule at all.

But enough with the bad news.

We had to take a close look at these very real problems so we could not only alert you to blind spots that can get your team in trouble, but so we could get serious about real solutions.

Three Steps to Better Sprint Planning and Execution with LiquidPlanner

LiquidPlanner was built to give teams a clear view of reality—including estimation uncertainty and the impact of change—that makes schedules reliable and sprint planning and execution better, easier, and more efficient. Here's how, in just three steps.

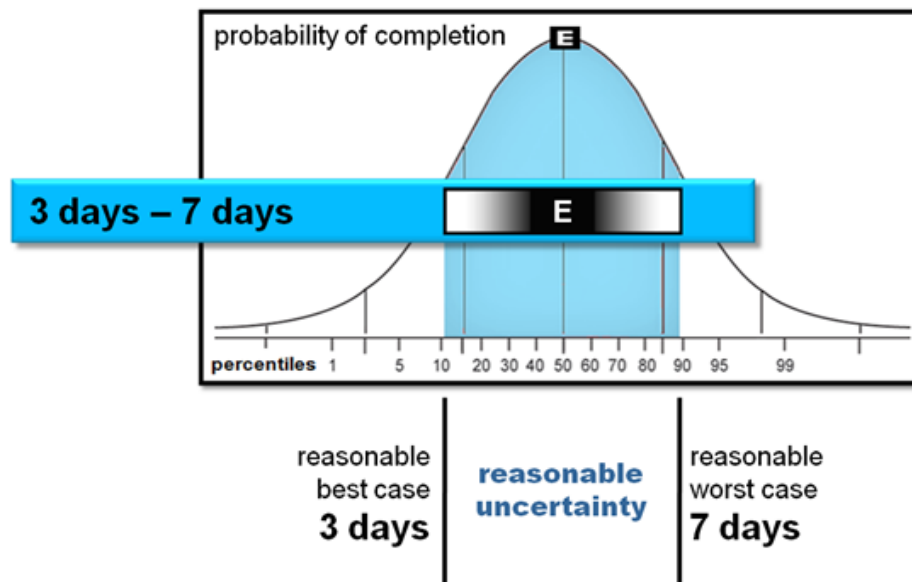
Step 1: Use estimates that reflect the way you actually work.

LiquidPlanner lets you estimate task completion in **ranged estimates of real time** (4-7 hours, for example). Not only is this the way you really think about time and work, it has a buffer of uncertainty built right in for everyone to see. It sounds simple, but the impact is profound.

With a ranged estimate you are giving a best-case (low end) and worst-case (high end) scenario of how long it will take you to complete the task. This range captures and conveys uncertainty; an estimate with a wide range, such as 2-5 days, conveys more uncertainty than a narrower estimate, such as 1-2 hours. As work progresses, the uncertainty narrows, and the schedule becomes an even more accurate reflection of reality.

Using ranged estimates also:

- Lets you plan accurately even with a large range of uncertainty
- Allows teams to drive uncertainty out of the schedule early (since it can be seen clearly)
- Prevents sandbagging
- Creates a neutral, honest point of communication within teams and with business owners



Step 2: Dynamically plan your sprint based on the amount of work (and uncertainty) in the tasks you want to complete.

A good way to explain sprint planning in LiquidPlanner is to tell you how we do it ourselves.

First, we create a new tasklist and label it with the number of the current sprint (we now have 20 successful sprints under our belt using LiquidPlanner). We work in four-week sprints, so we give that sprint a promise date that is four weeks out. Then we load up the sprint by entering new tasks or **dragging and dropping** tasks from the backlog. If any tasks have not yet been given an estimate, it's easy to do so right on the spot.

Once all tasks have been entered, LiquidPlanner's **probabilistic scheduling engine** immediately creates a schedule and lets us see if we will hit our target given the identified set of tasks. If the schedule shows we will end up late, it's just as easy to reprioritize and adapt the plan by dragging tasks in or out. Ranged estimates also provide a starting point for discussing and negotiating task estimates.

Prioritized Tasklists	[Remaining]	E days	12	Nov	Dec	Jan 2010
Sprint Planning	[53.95d - 67.3d]	60.63d				
Sprint 1	[20.82d - 26.93d]	23.88d				
Database support for email import (Agile Project > Database)	[2d - 5d]	3.5d				
Email import UI (Agile Project > Website features)	[3d - 5d]	4d				
Add new columns to customer record (Agile Project > Database)	[0.5d - 1d]	0.75d				
Change menu color scheme (Agile Project > Website features)	[0.5d - 1.5d]	1d				
Bug: calendar isn't updating (Agile Project > Bugs)	[0.25d - 1d]	0.63d				
Bug: right-click menu is broken (Agile Project > Bugs)	[1d - 2d]	1.5d				
Bug: sign-in message is wrong (Agile Project > Bugs)	[1d - 2d]	1.5d				
Website performance tuning (Agile Project > Operations)	[3d - 5d]	4d				
Firewall re-configuration (Agile Project > Operations)	[0.25d - 1d]	0.63d				
Sprint 1 testing (Agile Project > Testing)	[4d - 8d]	6d				
Website deployment (Agile Project > Operations)	[0.25d - 0.5d]	0.38d				
Sprint 2 (staging)	[19.82d - 31.18d]	25.5d				
Redesign website UI (Agile Project > Website features)	[5d - 15d]	10d				
Refactor database schema (Agile Project > Database)	[10d - 15d]	12.5d				
Bug: calendar control broken in FF3 (Agile Project > Bugs)	[2d - 4d]	3d				
Backlog	[9.54d - 12.96d]	11.25d				
New sign-in module (Agile Project > Website features)	[4d - 7d]	5.5d				
New sign-in module (database work) (Agile Project > Database)	[1d - 1.5d]	1.25d				
Investigate content distribution (Agile Project > Operations)	[1d - 2d]	1.5d				
Bug: change menu colors to blue (Agile Project > Bugs)	[0.5d - 1d]	0.75d				
Bug: internal error with long email addresses (Agile Project > Bugs)	[1d - 2d]	1.5d				
Bug: week calculations off by one (Agile Project > Bugs)	[0.5d - 1d]	0.75d				

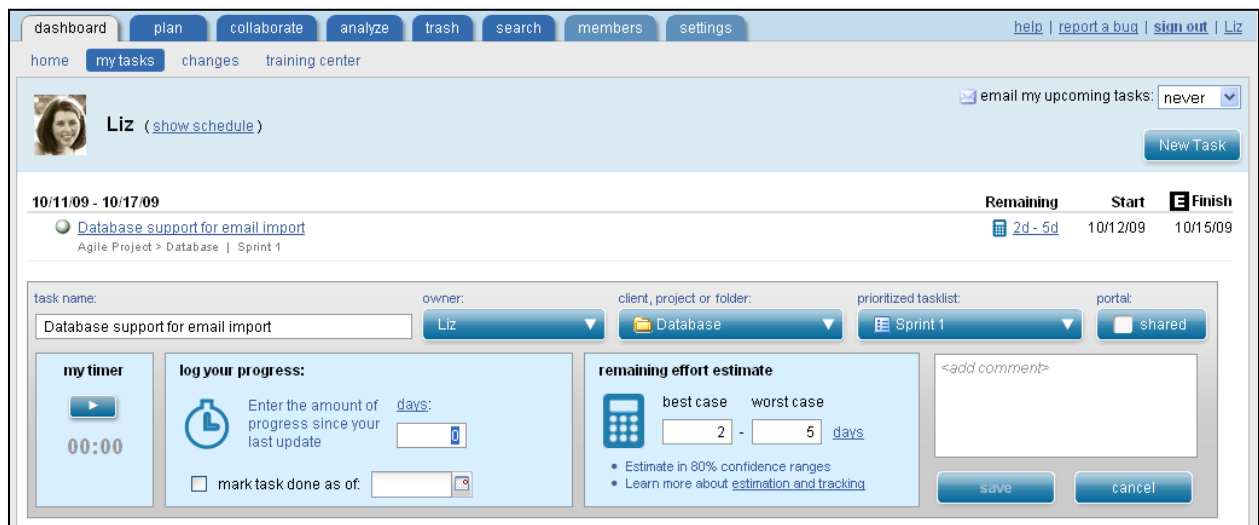
Step 3: Keep your schedule alive with frictionless updates and change management.

LiquidPlanner lets team members quickly and easily update progress and re-estimate their tasks right from their personal dashboard or iPhone. By focusing on re-estimation instead of an arbitrary-percent-complete status update, the schedule remains an accurate reflection of reality.

And since LiquidPlanner's scheduling engine recalculates every time a change is made (in real time), you can clearly see how statistically likely you are to hit your release date based on the actual progress of the team. If problems arise, you get a heads up to course correct long before your entire sprint is at risk.

When requirements change or new ones are introduced, drag and drop prioritization lets you visualize the impact of the change and **do "what if" planning on the spot**. Let's say that business owners want a new feature bumped up to the next sprint. By simply dragging that item into the next sprint, a new schedule is automatically created so you can make decisions and reprioritize other features as needed.

All of these features make it easy for individual team members to keep the schedule honest and alive. This frees managers up from playing babysitter and doing data input. And it means no deadly blind spots that could put the hard work of your talented team at risk.



Conclusion

Agile represents a move to more interactive, humanistic software development methods. While reliance on processes and tools is downplayed in favor of communication and collaboration, the demands placed on Agile teams requires more and better planning, estimation, and organization—not less. As such, Agile teams benefit from tools that appropriately support and enhance the way they work.

LiquidPlanner was not intentionally built just for Agile teams, and yet it intuitively embodies the Agile philosophy and supports the practical needs of Agile teams. It does so because it was developed by seasoned project managers looking for a solution to real problems, not because more project management software was needed in the world. For Agile teams looking to do what they do even better—deliver quality product on time with more ease and efficiency—take a look at LiquidPlanner.

About LiquidPlanner

LiquidPlanner is online project management software built on an innovative probabilistic scheduling system (patents pending) using Agile methodologies. It takes a new approach to frictionless collaboration and is designed to be flexible, intuitive, and smart enough to support fast-moving teams of all sizes and structures. LiquidPlanner, Inc. was founded in 2006 and is headquartered in Bellevue, WA.

About the Author

Jason Carlson co-founded LiquidPlanner with Charles Seybold in March 2006. Jason began his software career at Microsoft working as a Test Lead, focusing on performance and scalability of the MSN Expedia product. He then spent eight years at Expedia.com, first as a Software Test Manager and later as Director of Quality Management. Throughout his tenure, he was dedicated to ensuring the stability and scalability of Expedia's world-class internet travel platform. Jason has a Computer Science degree from Minnesota State University and is undeniably obsessed with maintaining a high standard of quality in every aspect of software development, from design, development and testing to customer support.

